DEGREE PROJECT IN ENGINEERING PHYSICS,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2021

Forward Modelling of Ground Based SST Telescope Images

ANNA HIDALGO LARSSON

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

KTH Engineering Sciences

Master's Thesis

Forward Modelling of Ground Based SST Telescope Images

Anna Hidalgo Larsson

Supervisors: Stefano Barra (SSC), Dr. Nickolay Ivchenko (KTH)
Examiner: Prof. Tomas Karlsson

Division of Space and Plasma Physics, Department of Electrical Engineering,
School of of Electrical Engineering and Computer Science,
KTH Royal Institute of Technology

Stockholm, Sweden 2021

Typeset in LaTeX

Examensarbete för avläggande av teknologie masterexamen i Teknisk fysik, inom ämnesområdet Flyg- och rymdteknik.

Master's thesis for the degree of Master in Science of Engineering (MSc), in the subject area of Aerospace Engineering.

TRITA-xxxx

Abstract

Space debris is becoming an increased threat to the future use of space orbits. In order to counteract this threat, the field of Space Situational Awareness (SSA), and the sub-field Space Surveillance and Tracking (SST), have been developed to gather knowledge about the space debris and satellites surrounding Earth. The orbit of a satellite can be determined by acquiring images of the satellite using a telescope and a sensor. During this thesis, a tool has been programmed in Python. This tool can simulate these types of images of satellite passes, at a given time and location. The simulator takes the system parameters of the telescope and camera sensor into account, together with several different types of disturbances which affect these images. The project has been carried out at the Swedish Space Corporation (SSC), which recently launched an SSA initiative. They plan to use these images to learn more about their upcoming observations, and possibly to test an orbit determination software.

Keywords: SST, SSA, Satellite, Telescope, CMOS, TLE, Image Simulator, Space Imagery, Space Debris, Tracklet

Sammanfattning

Rymdskrot är ett allt mer påtagligt hot mot den framtida användningen av omloppsbanor i rymden. För att motverka detta hot har det blivit viktigt att kartlägga rymdlägesbilden och de objekt som ligger i omloppsbana runt jorden. Detta görs genom att observera, identifiera och banbestämma satelliter. En satellits omloppsbana kan bestämmas genom att ta bilder av satelliten med hjälp av ett teleskop och en sensor. Under detta examensarbete har ett verktyg för att kunna simulera sådana bilder utvecklats. Simuleringsverktyget har programmerats i Python och kan simulera bilder av satellitpass vid en given tidpunkt och från en given plats. Verktyget tar hänsyn till systemparametrarna för teleskopet och kamerasensorn, samt effekterna av ett flertal olika typer av störningar som påverkar dessa bilder. Projektet har genomförts hos företaget Swedish Space Corporation (SSC), som nyligen lanserade ett initiativ för att bättre förstå rymdlägesbilden. De planerar att använda dessa bilder för att lära sig mer om deras kommande observationer, samt att eventuellt testa en programvara för att bestämma banparametrar.

Nyckelord: Rymdlägesbild, satellit, teleskop, CMOS, TLE, bildsimulator, rymdbilder, rymdskrot, satellitpass.

Preface

This master's thesis is my final work at the Royal Institute of Technology in Stockholm, Sweden. With it, I will have completed the degree programme in Engineering Physics, which awards the Swedish title *civilingenjör*. It is also the last part of the master's programme in Aerospace Engineering, where I have specialised in Space Technology.

Space has been a major interest of mine for more than ten years. It is an area of science I find fascinating in many ways, as well as an area that many people have some kind of relation to. The combination of innovation, exploration, public engagement, and setting new boundaries for what is possible is something I have not yet witnessed in any other field of science. Thanks to this interest for space exploration, I have had the opportunity to visit some of the best telescopes in the world in Chile, and on La Palma, Spain. I have also had the pleasure to visit multiple satellite stations. It has therefore been a very interesting project to model many of the complicated technologies associated with these areas of exploration.

Today, many people use satellite technologies daily without being aware of it. The services provided by satellites have become crucial to the modern society. However, space debris is becoming an increased threat towards the future use of orbits, and therefore also satellites. This threat requires a great deal of work to be able to ensure sustainable usage of space orbits in the future. I hope that this thesis might serve as a contribution to this important work.

Acknowledgements

I would like to extend my gratitude to my SSC supervisor Stefano Barra. We have spent many hours discussing encountered problems and their possible solutions. He has always been eager to help, guide, and encourage me throughout this project. He has also always been quick to reply to my questions and meeting requests. These are valuable abilities of a supervisor. I would also like to thank Jacob Ask from SSC who has also been very much engaged in the thesis project. His ideas and ability to find the right expertise when needed helped making this an even better project. I would also like to offer my gratitude to Anton Bjöörn at SSC, who happily answered all my questions related to programming and IT. I wish all of them the best of luck in their upcoming work related to the SSA initiative.

From KTH, I want to thank my academic supervisor Dr. Nickolay Ivchenko for the fruitful discussions and advise about the project. Additionally, I want to thank my examiner Prof. Tomas Karlsson for taking on the examining task.

During the project I talked to multiple astronomers regarding some points of the work. I would especially like to thank Dr. Alexis Brandeker, Dr. Flavio Calvio, and Alexander Pietrow for taking their time to discuss aspects of the work.

During my years of studying, I have been very lucky to always have many people who have supported and encouraged me throughout my successes and encountered hardships. First of all, I would like to thank my mum and dad for their never-ending support. An equally big thank you is directed towards my dear brother and sister, who helped me stay motivated through trying periods.

My time at KTH has also included meeting some of my closest friends. These years would not have been as enjoyable without out them, and I very much want to thank Anders, Hilda, Thomas, and Josefine for this. I would also like to especially thank Anders and Thomas who always were willing to discuss different topics and problems of the thesis project. Additionally, I want to thank my dear friend Pernilla for being there for me long before and throughout the whole KTH experience.

Last of all, I want to thank Philip for being at my side everyday throughout this project. I am forever grateful for your support, encouragement, love, and help. This project would not have been equally good without you.

This work has made use of data from the European Space Agency (ESA) mission *Gaia* (`https://www.cosmos.esa.int/gaia`), processed by the *Gaia* Data Processing and Analysis Consortium (DPAC, `https://www.cosmos.esa.int/web/gaia/dpac/consortium`). Funding for the DPAC has been provided by national institutions, in particular the institutions participating in the *Gaia* Multilateral Agreement.

Contents

List of Figures

List of Tables

Nomenclature

Acronyms

ADC	Analog-to-Digital Circuitry
ADQL	Astronomical Data Query Language
AIUB	Astronomical Institute of the University of Bern
ASA	Astro Systeme Austria
CCD	Charge-Coupled Device
CMOS	Complementary Metal–Oxide Semiconductor
DEC	Declination
DPAC	Gaia Data Processing and Analysis Consortium
ESA	European Space Agency
FFT	Fast Fourier Transform
FITS	Flexible Image Transport System
FOV	Field of View
FPS	Frames per second
GEDR3	Gaia Early Data Release 3
GEO	Geostationary Orbit
GSOC	German Space Operation Center
GUI	Graphical User Interface
IFOV	Instantaneous Field of View
ISS	International Space Station

JD	Julian Date
JPL	Jet Propulsion Laboratory
LEO	Low Earth Orbit
MSL	Meters above Sea Level
NASA	National Aeronautics and Space Administration
PNG	Portable Network Graphics
PSF	Point Spread Function
RA	Right ascension
RAAN	Right Ascension of the Ascending Node
RCS	Radar cross section
RMS	Root mean square
SAAO	South African Astronomical Observatory
SGP4	Simplified General Perturbations
SMARTnet™	Small Aperture Robotic Telescope Network
SNR	Signal Noise Ratio
SSA	Space Situational Awareness
SSC	Swedish Space Corporation
SST	Space Surveillance and Tracking
STK	Systems Tool Kit
TIFF	Tag Image File Format
TLE	Two-Line Element set
UTC	Coordinated Universal Time

Symbols

α	Right ascension
δ	Declination
$\oplus$	Earth
λ	Wavelength

ν	True anomaly
Ω	Longitude of the ascending node
ω	Argument of periapsis
ϕ	Phase angle
σ	Radar cross section
σ_{SD}	Standard deviation
$\odot$	Sun
τ	Efficiency
θ	Diffraction limit
a	Semimajor axis
a_0	Albedo
D	Diameter
e	Eccentricity
f-ratio	Focal ratio
i	Inclination
L	Focal length
M	Absolute magnitude
m	Apparent magnitude
N	Focal number
n_p	Number of pixels
p	Pixel size
Q_{eff}	Quantum Efficiency
S	Photon flux
B	Byte
rad	Radians

Chapter 1

Introduction

1.1 Space Situational Awareness and its sub-fields

In 1957, the Russian satellite Sputnik 1 was launched and the space age commenced. Since then, more than 10,000 satellites have been placed into orbit around Earth, and of these 6,250 satellites still remain there [1]. However, today only 3,600 satellites in Earth orbit remain functional while the rest of the satellites have become space debris. The definition of space debris is a non-functional, man-made object in space [2]. Besides non-functional satellites, the space debris also includes large resident space objects such as spent upper rocket stages, down to small objects from explosions and collisions such as paint flakes and fragments. The Space Surveillance Network regularly tracks 28,210 space debris objects of different sizes [1]. Many of these objects are populating the same regions in space as operational satellites, with altitudes ranging between 300 to 40,000 km [2]. The current space debris in orbit thus pose a risk of causing new collisions, increasing the number of debris objects even further. The worst case would be a scenario where the Kessler syndrome is triggered, meaning an exponential increase of space debris by a chain reaction of collisions and explosions [3]. This could make spaceflight too dangerous to continue pursuing, and it is therefore in the interest of all space stakeholders to avoid these types of scenarios.

In February 2009, a collision between an active Iridium communication satellite and a non-functional Russian satellite occurred unexpectedly. This was the spark of what has become known as Space Situational Awareness (SSA)[4]. The definition of SSA varies, but according to Kennewell and Vo (2013), SSA is in its broadest sense defined as the knowledge of the energy and particle fluxes in the near-Earth space environment, together with the natural and artificial objects passing through or orbiting within this space. It also includes the past, present and future state of these particles and objects.

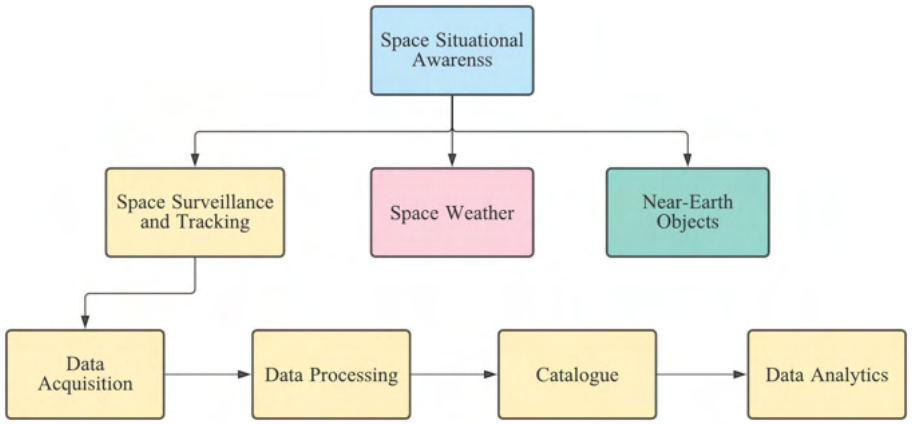


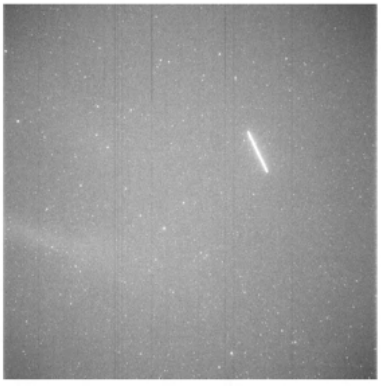
Figure 1.1: Division of SSA into different sub-fields.

The European Space Agency (ESA) has divided their SSA Programme into three main areas: Space Weather, Near-Earth Objects, and Space Surveillance and Tracking (SST) which is defined as watching for active and inactive satellites, discarded launch stages and fragmentation debris orbiting Earth [5]. SST information therefore provides the ability to detect and predict the movement of space debris in orbit around Earth, and can be used by spacecraft operators to avoid collisions. This information is provided by data acquisition and processing, and is stored in a database referred to as a catalogue. The division of SSA into different sub-fields is illustrated in Figure 1.1.

Raw data of resident space objects is acquired by having sensors scanning large areas of the sky for moving objects. Different types of sensors can be used for this purpose, but radar and optical sensors are common. Optical data is collected at an optical observatory, which consists of a telescope, a camera, a telescope mount, and a dome. The data is then processed into a tracklet which is a fragment of the track followed by the object, examples of tracklets can be seen in Figure 1.2. The tracklet is converted into an ephemeris consisting of the position and velocity of the object, which is catalogued. Data is used for catalogue build-up, meaning detection of new objects without any prior information. Usually, a single tracklet is not reliable enough to estimate the orbit of the object, meaning track association becomes mandatory [6]. New data measurements are continuously required in order to maintain the catalogue and keep it updated.

1.2 Optical observations and orbital extraction

Both radar and optical sensors can be used to to detect space debris, but optical sensors benefit of higher sensitivity for the detection of objects at large distances [2]. For optical observations to be possible, the sky must be dark while the object is illuminated. For objects orbiting Earth in Low Earth Orbit (LEO), this happens around 1-2 hours after sunset and before sunrise, depending on the latitude and the season. Images can be taken by either using sidereal or satellite tracking. If the telescope uses sidereal tracking it tracks the background stars on the sky and the satellite is shown as a tracklet on the resulting image. When using satellite tracking for observing known satellites, the stars instead are shown as tracklets. The difference between these modes are shown in Figure 1.2. The length of the tracklets depend on the exposure time and object's orbital speed.



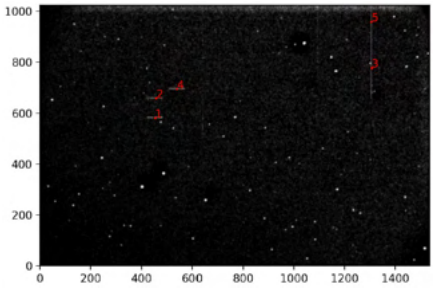
(a) Sidereal tracking showing the tracklet from Tiangong-1.

(b) Satellite tracking of two geostationary satellites marked by arrows.

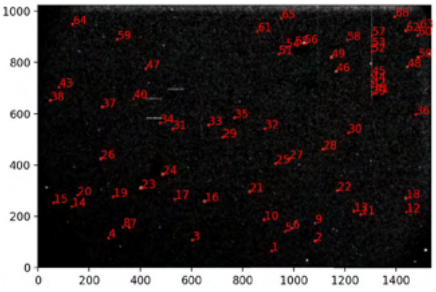
Figure 1.2: Sidereal tracking versus satellite tracking. By tracking the stars the satellite is shown as a tracklet in the resulting image, and vice versa for satellite tracking [7].

The images are then processed and undergo a noise removal procedure. The process of extracting the orbital parameters of the satellite vary, but one approach is described in S. H. Hossein et al. (2020), where an image captured by sidereal tracking is analysed [8]. After image noise removal, the sky survey algorithm uses the Canny edge identification algorithm and identifies different objects in the image as stars or tracklets by analysing each pixel. The inertial moments of these objects are calculated together with the center of mass light curves, which are used to identify stars. An example of target and star identification is seen in Figure 1.3a and Figure 1.3b. If stars are found, the algorithm proceeds with computing the center of mass of the stars, together with the image and a catalogue of triangles index. An example of the resulting triangles is seen in Figure 1.3c. The triangles

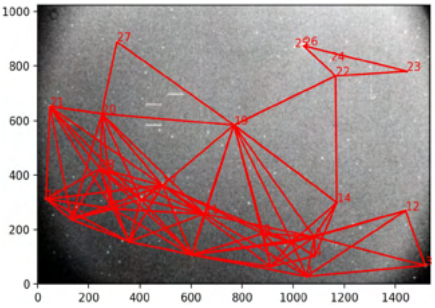
are matched to a catalogue which allows for retrieval of the celestial coordinates. It is then possible to extract the orbital parameters of the satellite trajectory in a format called a Two-Line Element set (TLE).



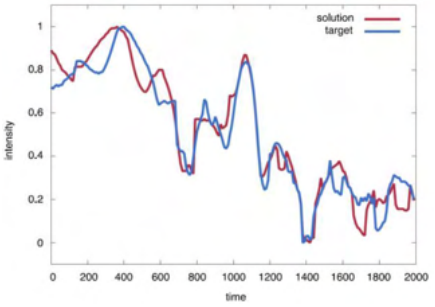
(a) Identified targets.



(b) Identified stars.



(c) Identified triangles.



(d) Analysis of the tracklet luminosity.

Figure 1.3: The different steps when processing an image for orbit and attitude determination [7].

The orbital determination estimation can be improved by multiple observations, but these images can also be used for attitude determination [8]. This is done by identifying the variation of light flux of the identified satellite, which is used to predict the attitude with respect to the velocity vector direction. An example of a light curve generated from a tracklet luminosity analysis is seen in Figure 1.3d.

1.3 Optical system properties and disturbances

The imagery described in the previous section can be affected by several factors, such as telescope properties and external disturbances. These can diminish the accuracy and precision of the object's orbit estimation. It is therefore important to examine in detail how these factors affect the imagery.

An optical system has different abilities depending on its properties such as aperture, focal length, field of view and exposure time for example. These will

determine how much of the sky the telescope can see, and also how faint objects the telescope can discern.

The images are taken via a sensor, usually either a charge-coupled device (CCD) or a complementary metal-oxide semiconductor (CMOS). Some examples of sensor properties affecting the image is shutter type, read out times, duty cycle, quantum efficiency, and full-well capacity [9]. These sensors are also prone to certain disturbances such as read noise, dark currents, and dark and hot pixels.

The imagery can also be affected by a number of external disturbances, for instance vibrations due to passing vehicles close to the observation site. Other examples are mounting vibrations, moonglow, skyglow and atmospheric effects. Operational parameters such as binning may also have effects on the imagery and subsequent orbit determination.

1.4 Purpose and objectives

The Swedish Space Corporation (SSC), formerly known as Rymdbolaget, is a mainly commercial space company owned by the Swedish government. SSC operates Esrange Space Center in northern Sweden where they carry out sounding rocket and balloon missions. The company also provides science and launch services, as well as satellite ground network services. Currently, SSC is deploying an optical SST station to build capabilities as part of an SSA initiative.

The objective of this thesis is to design and develop a tool that SSC can use to simulate images of satellite passes in the night sky, at a given time and location provided by coordinates, as detected from a professional optical telescope system. This will help determining several feasible observational scenarios that will be used to define the telescope operations. It will also allow SSC to investigate how system properties and external disturbances affect the image quality of an optical SST telescope. The satellite orbits are obtained by propagating a TLE, in order to simulate satellite tracklets on the simulated sky. However, the tool should also provide the possibility to switch between sidereal and satellite tracking depending on the user's needs. The simulator shall also be able to model some disturbances, i.e. atmospheric effects, and these effects shall be applied to the simulated image. SSC also plans to use these images to train and test an orbit determination software.

The implementation of the simulation tool is done by designing, programming, and testing a tool based in Python. When suitable, pre-existing tools, libraries, and packages should be used.

1.5 Delimitations

This thesis aims at developing a space imagery simulation tool, and in order to make this feasible within the given time period an important strategic decision has been to make use of already existing code libraries when possible. Developing code

to model i.e. the propagation of satellite trajectories, is therefore not included in the scope of the project. Where preexisting code modules are not found to exist already, necessary code will be developed.

In reality, there are many types of disturbances that might affect the images to be simulated. This thesis work does not include modelling minor effects such as those from gamma rays or vibrations caused by external events, such as nearby vehicles for example. The type of disturbances the images are prone to are often complicated phenomena, and simplifications are therefore applied where necessary.

TLEs will be used to simulate the satellite orbits on the simulated sky, but this thesis does not include the inverse modelling of extracting TLEs from the simulated images. This process is complex, and extensive work has already been performed by others.

The visibility of a satellite depends on many factors, i.e. area, albedo, geometric shape, spin rate and solar phase angle to mention some of them. These many factors make the modelling of satellite magnitude quite complicated, and thus simplifications are used in this process.

1.6 Structure of the thesis

Chapter 2 presents the relevant background information of optical systems, cameras and detectors, satellites, orbits, and coordinate systems. It also familiarises the reader with different types of disturbances which affects SST images.

Chapter 3 presents the system requirements and system architecture of the tool. It then describes the process of developing the simulation tool, and the different problems encountered during the development. The chapter also presents different assumptions used in the modelling of the images, and gives the motivations behind different choices of modelling and estimations.

Chapter 4 presents the results from running the simulation tool. These results are based on trying two reconstruct two real SST images. The chapter also summarises which packages the tool depends on, evaluates the initial requirements, and summarises possible future work.

Chapter 5 summarises the report and presents the conclusion from the work. It also summarises the limitations of the simulation tool.

Chapter 2

Background

This chapter describes the background information of optical systems, cameras and detectors, sensor imaging, visibility of satellites, the night sky, orbits and the sources of different image disturbances. Current software tools used in astronomy and orbital determination are also outlined. The chapter also gives the necessary theory and terminology crucial for the understanding the work. The equations presented in this section are used in the simulation tool to calculate optical system properties and other parameters.

2.1 Optical systems

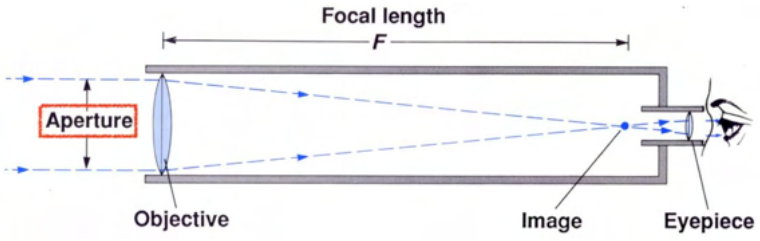
In 1609, the Italian physicist and astronomer Galileo Galilei became the first person to build a telescope and use it for astronomical observations [10]. Since then the telescope has undergone major development in order to enable observations of fainter objects. Unlike other areas of science, astronomy is limited to observations and the purpose of an optical telescope is to collect as much light as possible. In this section the function of a telescope is described and properties of a telescope are explained.

2.1.1 Different types of telescopes

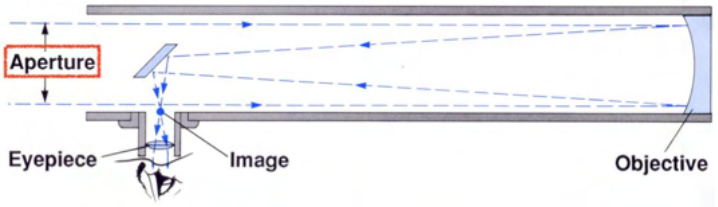
An optical telescope gathers light by either a lens or a mirror, and this difference classifies the telescope as either a *refractor* (lens) seen in Figure 2.1a, or a *reflector* (mirror) seen in Figure 2.1b [11]. They are used for the same main purpose: gather light from a large area to study faint sources, but the difference in the geometrical optics result in different areas of usability.

Refractors

A refractor uses a lens called an objective to collect incoming light. The refractor can also be equipped with an eyepiece and the distance between it and the focal plane is adjusted to get the image into focus. A glass lens refracts different colours differently, which means the colours do not meet at the same focal plane. This is known as *chromatic aberration* and is corrected by using multiple lenses and different glasses. The size of the glass lens limits the possible aperture, and refractors are limited by a small field of view and a long structure. However, they are still used for some applications such as solar telescopes for example.



(a) Schematic of a refractor telescope.



(b) Schematic of a reflector telescope.

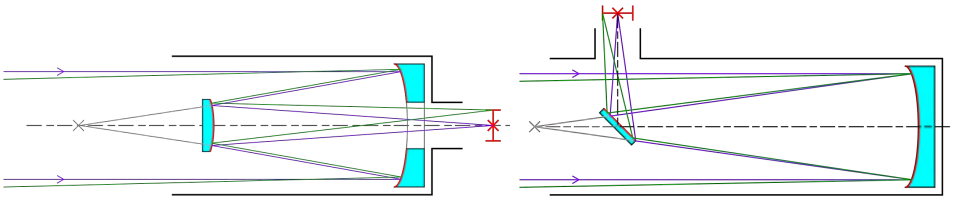
Figure 2.1: Schematics of two telescope types [12].

Reflectors

A reflector telescope uses a mirror to gather incoming light, and reflectors are the most commonly used telescope type for research purposes. The mirror is usually coated with a thin layer of aluminium and is of concave shape which allows it to focus all incoming light into the same focal point. Thus, reflectors do not suffer from chromatic aberrations [11]. However, the reflector can suffer from aberrations such as *spherical aberration* and *coma* due to that the light rays do not converge at the same point in the focal plane [10].

Reflectors are divided into different types depending on its focus. The first reflector was a *Newton focus* telescope which guides the light from the primary

mirror by using a secondary small, flat mirror [11]. Such a reflector is seen is in Figure 2.2a. The second type of reflector is called *Cassegrain focus*. In such a telescope the primary mirror is parabolic and the rays are then reflected by a secondary small, hyperbolic mirror through a hole in the main mirror [11]. A telescope of Cassegrain type is seen in Figure 2.2b.



(a) Newtonian telescope. [13] **(b)** Cassegrain telescope. [14].

Figure 2.2: Path of light through two different telescopes of reflector type.

A variant of the Cassegrain telescope is the *Ritchey-Crétien* telescope where both mirrors are hyperbolic in order to eliminate coma [11]. Many large telescopes such as the Hubble Space Telescope is of the Ritchey-Crétien type. Another variant is the *Corrected Dall-Kirkham* telescope where also both mirrors are hyperbolic, but it also utilizes corrector lenses along the optical path [15].

2.1.2 Telescope properties

A telescope has a large field of application which varies depending on its properties. In this subsection these properties are explained.

Aperture

In an optical system, an aperture is the size of the first optical element, such as the diameter of the main lens or mirror [10]. The aperture diameter is the simplest measure of how much light the telescope can gather. A larger aperture means a greater number of photons are intercepted by the telescope, thereby increasing the sensitivity of the optical system. The apertures of a refractor and a reflector telescopes has been marked in Figure 2.1.

Coatings and interference filters

Refractive optics use anti-reflection coatings to eliminate reflections between different components in the telescope [16]. The thickness of the layer is dependent on wavelength, and therefore limits the perfect functionality of the telescope to a certain wavelength. For other wavelengths, losses will occur due to transmission and reflectance factors.

Diffraction limit

When a light wave encounters an obstacle or an opening, the light "bends around the corner" and deviates from rectilinear propagation [10]. This phenomenon is a characteristic of the wave nature of light. The theoretical limit for the resolution of the telescope is set by the diffraction of light [11].

The diffraction limit, θ, is the finest detail an aberration-free optical system can distinguish when atmospheric turbulence is not present [10]. It is therefore the limit of the maximum achievable resolution of the telescope, beyond this limit the image becomes blurry. Incoming light gets diffracted when passing the aperture, and the limit is therefore a function of the size and shape of the aperture and the wavelength of the incoming light. It can be calculated as

$$\theta = \frac{1.220 \times \lambda}{D} \qquad [\text{rad}] \tag{2.1}$$

where λ is the wavelength of the incoming light and D is the diameter of the telescope [17].

Diffraction pattern from a circular aperture; the Airy disk

A star seen from Earth is a point source of light. When a point source is observed by a long exposure, the light will be "smeared" by the effects of the telescope optics, the atmosphere, and other disturbances [18]. Assuming the optics are free from aberrations, the light will be distributed by the characteristic Point Spread Function (PSF) which describes the variation of the intensity with distance from the center of an image of a point source created by the optical system [10]. The contribution to the PSF from an optical system with a circular aperture is called the Airy disk, and it is illustrated in Figure 2.3. The circular aperture diffracts the image of the point source into a very bright central spot, surrounded by concentric bright rings, whose brightness decreases with the distance from the center [19]. The angular radius of the first dark ring is equal to the diffraction limit θ, in accordance with Rayleigh's criterion for the resolution of a mirror or lens [17].

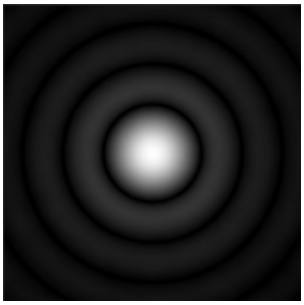


Figure 2.3: The Airy disk, showing the bright central spot surrounded by concentric bright rings [20].

Diffraction spikes

Reflecting telescopes require a secondary mirror that needs to be attached and supported by the telescope. This is done by using stiff, supporting vanes that can be arranged in a few different strut arrangements. Some possible arrangements are illustrated in Figure 2.4, together with their diffraction patterns which happens due to the vanes diffracting the light [10]. These patterns can be seen as "spikes" in stellar images. Each straight obstruction in the beam produces two diffraction spikes 180° apart in the direction perpendicular to the direction of the vane. Diffraction spikes also contribute to the PSF of the optical system.

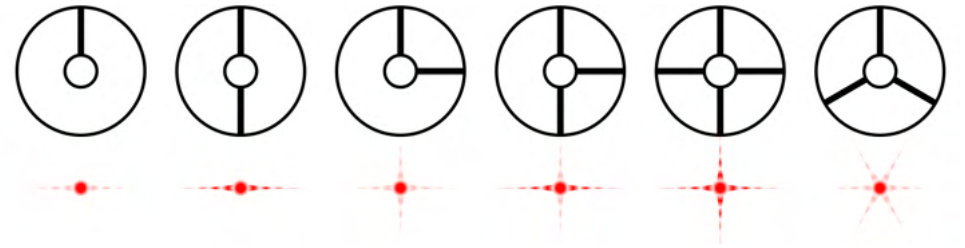


Figure 2.4: Comparison of diffraction spikes for various strut arrangements of a reflecting telescope. The inner circle represents the secondary mirror which is supported by vanes [21].

Focal length

The focal length of an optical system is the distance from the point where light rays converge to form a sharp image of an object to the digital sensor or focal plane, when the lens is focused at infinity [22]. The focal length is important for the field of view of the optical system. A longer focal length means a narrower field of view with an equivalent sensor size, and similarly a shorter focal length indicates a wider field of view. The focal length of a refractor telescope is highlighted in Figure 2.1a.

Focal ratio and focal number

The focal ratio (f-ratio) is the ratio of the effective focal length of an optical system to the diameter of the aperture [10]. It is a useful parameter to classify and compare different telescopes, and it is calculated as

$$f/N = \frac{L}{D} \tag{2.2}$$

where N is the focal number, L is the focal length and D the aperture diameter.

Field of view

The field of view (FOV) describes how much of the sky the telescope can see in one frame. It is related to the instantaneous field of view (IFOV) which describes the solid angle through which a pixel can receive electromagnetic radiation. For SST purposes, the optical system is combined of a telescope and an imaging sensor. The IFOV for a pixel can be calculated as

$$\text{IFOV}_{\text{pixel}} = 2 \arctan\left(\frac{p}{2ND}\right) \qquad [\text{rad}] \tag{2.3}$$

where p is the pixel size, N is the focal number and D the aperture diameter [23]. The equation is originally given with the unit in steradian, but since equation (20) in [23] is for $\text{IFOV}^2_{\text{pixel}}$, and equation (2.3) is for $\text{IFOV}_{\text{pixel}}$, the unit becomes radians. The FOV for the imaging sensor can then be calculated as

$$\text{FOV}_{\text{sensor}} = n_p \text{IFOV}_{\text{pixel}} = 2n_p \arctan\left(\frac{p}{2ND}\right) \qquad [\text{rad}] \tag{2.4}$$

where n_p is the number of pixels in one side of the sensor. If the sensor is not a square sensor it is therefore necessary to calculate two FOVs to fully understand what the system can observe.

2.1.3 Telescope Mounting

There are two principal ways of mounting a telescope: *equatorial* and *altazimuthal* mounting. These types of mountings are based on two different coordinate systems which are further described in Section 2.4.2. A telescope is required to be mounted on a steady support and be able to rotate smoothly in order to prevent shaking and causing disturbances during observation, as well as to provide precise and accurate pointing [11]. An example of the zigzagged tracklet produced by mount vibration disturbances is seen in Figure 2.5.

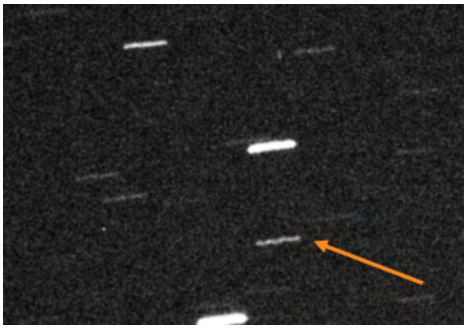


Figure 2.5: Example of mount vibrations disturbance in a satellite tracking image [24]. Mount vibrations cause the star tracklet to look zigzagged.

Equatorial mounting

Equatorial mounting, seen in Figure 2.6, compensates for Earth's rotation by having one axis called the *polar axis* or *hour axis* directed towards the celestial pole [11]. This axis is therefore parallel to the axis of the Earth, and turning the telescope around it at a constant rate compensates for the apparent rotation of the sky. The other axis is parallel to the hour axis and is called the *declination axis*. The equatorial mounting can cause heavy loads on the bearings of the telescope.

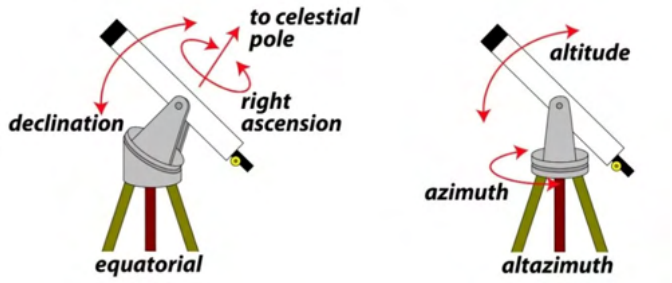


Figure 2.6: Equatorial and altazimuthal mount of a telescope [25].

Altazimuthal mounting

When a telescope is mounted using altazimuthal mounting, one axis is vertical and the other horizontal [11]. This is also illustrated in Figure 2.6. To be able to follow the apparent rotation of the sky the telescope must be turned about both axes with changing velocities. This makes the controlling of it more complicated and usually requires a computer to handle. Altazimuthal mounting is easier to construct than equatorial mounting, and is more stable for large telescopes. Close to zenith there is a small forbidden region where the telescope cannot observe since it would cause a quick change of the azimuth by 180°.

2.2 Cameras and detectors

To capture the image the telescope produces it is necessary to use a detector. The role of the detector is to detect, as noiselessly and efficiently as possible, each precious photon collected by the telescope and instrument [10]. In the optical range it is most common to use a detector based on semiconductor technology. The two types that are in use today are the CCD and CMOS sensors.

This section covers the functionality of imaging sensors and the different parameters that categorise them. It also presents the operational parameters, and why one detector is preferred for observing orbital debris. It then continues with how

images from the sensors are stored digitally, before finishing with discussing how to determine the visibility based on signal noise ratio from the detector.

2.2.1 CCD detectors

A CCD is a two-dimensional array of p/n junctions made of silicon. Incident light on the semiconductor that falls within the bandgap of the photosensitive material produces electron-hole pairs [9]. The electrons are then trapped in potential wells produced by numerous small electrodes, and there they accumulate until their total number is read out by charge coupling the detecting electrodes to a single readout electrode [17]. This means that the electron-hole pairs are converted to digital counts by some sequence of amplifiers and analog-to-digital circuitry (ADC) [9]. The process of charge coupling consists of cycling the voltage of the electrodes and transfer the charge between them [17]. The charge is transferred through the whole structure of electrodes by continuous cycling until it is brought to the output electrode from where the charge value can be determined.

Until the readout process of the CCD begins, the photoelectrons generated on the sensor are restricted to specific spatial locations due to the bias placed on the surrounding readout electrodes [9]. The charge is then read vertically through adjacent pixels into a row, before being read horizontally to the amplifier and ADC, as seen to the left in Figure 2.7. Due to serially reading of the pixels, this process can give raise to unwanted artifacts and long readout times for large CCDs. An example of an artifact is *blooming*, which occurs when the charge in a pixel exceeds the saturation level and starts to fill adjacent pixels [26]. CCD sensors are typically designed to allow easy vertical shifting of the charge while potential barriers are created to reduce flow into horizontal pixels. The result is that excess charge prefers flowing into the nearest vertical neighbours, and blooming therefore produces a vertical streak in the resulting image.

The front side of the CCD is partially obscured by metal electrodes [10]. This implies that although the CCDs can be illuminated from either side it is preferred in astronomical applications to use *backside illumination*. The placement of the electric wiring then allows for more electrons to reach the sensor. The disadvantage of backside illumination is that it can result in poor sensitivity to some wavelengths.

2.2.2 CMOS detectors

The CMOS detector technology shares many of the same detection principles with the more traditional CCD technology. The main difference is that CMOS technology places the readout circuitry predominantly on each pixel [9]. This difference is illustrated in Figure 2.7. This is accomplished by having each pixel independent from adjacent pixels, instead each pixel converts its charge into an amplified voltage. This type of readout process decreases readout times substantially.

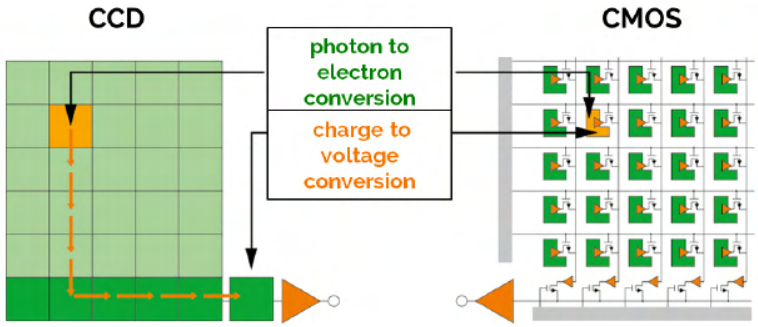


Figure 2.7: Readout architectures of a CCD and a CMOS sensor [9].

2.2.3 Detector parameters

Image sensors are assessed and compared by several parameters. The most important parameters are summarised in Table 2.1, and some of them are further elaborated upon.

Dark and hot pixels

Dark pixels (black pixels) are individual pixels or small clusters of pixels of the sensor which have significantly lower response than their neighbours (less than 75% the response of the average pixel) [28]. They usually form over time due to contamination of the sensor, but their effects on the image can be removed by applying a flat field reference or post processing interpolation.

Hot pixels are the opposite of dark pixels, these are pixels with a much higher dark current than their neighbours (50 times higher than specification) [28]. Similarly as for dark pixels they form due to contamination. Their effects can be removed by taking a background image with the shutter closed, a process which is called *dark calibration*.

Dark current

Dark current is the current measured when no light is falling on the detector [10]. This is due to the fact that the sensor cannot distinguish between photons from light or photons generated by heat [27]. The dark current signal increases linearly with exposure time, but can be measured by taking exposures with the shutter closed. Subtracting the dark current from the observed image then gives the real number of electrons due to incident light [11]. Individual pixels may vary in sensitivity, but this can be corrected for by taking a *flat-field* which is an image of an evenly illuminated field where the stars are not visible. Observations are then divided by the flat-field to remove the error caused by different pixels. Dark current is

Table 2.1: Parameters used for assessment and comparison of detectors.

Parameter	Explanation
λ_m	The wavelength for which the detectivity is maximum [17].
Dark and hot pixels	Non-functional pixels.
Dark current	The current measured from the detector when it is unilluminated [10]. Measured in electrons per second.
Dynamic range	Ratio of the largest to the smallest signal level a circuit or detector can handle [10]. Determines the range over which the sensor can simultaneously record very low and very bright signals [27].
Fill factor	Ratio of light sensitive area to the total area of the sensor.
Max. frame rate	The maximal rate of how often the sensor can capture an image, usually expressed as frames per second (FPS).
Pixel count	Number of pixels in the sensor.
Pixel size	Lengths of the sides of a pixel.
Quantum efficiency	Ratio of the actual number of photons that are detected to the number of incident photons [17]. Usually given as a percentage.
Read noise	The noise produced at each pixel when the sensor is read, expressed in number of electrons.
Shutter type	Global or rolling for electronic shutters.
Well depth	The limit on the number of electrons a pixel can hold before saturation [27].

temperature dependent and can therefore also be partly avoided by cooling the sensor to very low temperatures.

Pixel size

Pixel size is the length of the sides of a pixel and it is measured in meter. For observations it is important to know how much of the sky each pixel covers, and this can be adjusted by changing the optics on the telescope and camera [10]. The pixel size can affect other parameters such as the optical system resolution, dark current, readout noise, blooming, dynamic range, and sensitivity to cosmic rays.

Quantum efficiency curve

Quantum efficiency (Q_{eff}) is defined as the ratio or percentage of photons converted into a signal [27]. The sensor is characterised by a quantum efficiency curve which describes the necessary amount of electrons to generate a detectable signal at different wavelengths. To make the best use of the sensor the optics must be adapted for the wavelengths with highest quantum efficiency.

Readout noise

When the signal collected on CCD pixels is transferred, amplified, and converted to a digital value, noise is introduced at each step of the process [10]. Reading the voltage at the output node has an associated noise level which is independent of the actual number of electrons, including the case of no electrons [27]. This noise, which is added from each pixel during signal readout, is called readout noise. The readout noise is independent of exposure time [18].

The CCD readout noise can be described by a single readout noise value since all pixels pass through a common architecture and are subjected to the same sources of noise during the readout process [29]. In the CMOS on the other hand, each pixel has its own amplifier circuit resulting in slightly different read noise values resulting in a noise distribution. The noise level for the CMOS sensor is therefore usually given as a median value.

Shutter type

The shutter of the detector can be either mechanical or electronic. There are two types of electronic shutters: rolling and global. The rolling shutter exposure time is defined as the exposure time per row [9]. This means that the exposure of each row is offset from the adjacent row leading to pixels being read at different times which can cause spatial distortion of fast moving objects. They are therefore not optimal for observing space debris due to the required accurate timing. Using a global shutter means that every pixel on the sensor is exposed and read at the same time, and with the same exposure time. However, this mode can decrease the frame rate compared to rolling shutters and allows for higher read noise [29].

In a CMOS camera the electronic shutters can operate in overlapping modes [9]. This means that the readout of the previous frame is performed during the exposure of the current frame, and duty cycles of 100% is therefore possible.

2.2.4 Operational parameters

The image sensors are also affected by how they are operated. These operational parameters are summarised in Table 2.2 and some of them are elaborated upon further.

Exposure time

The exposure time affects the amount of light captured by the detector. The signal and noise grow with increasing exposure time [18]. Depending on the purpose of the observation and equipment, the exposure time can range between milliseconds and hours, or even days [10, 17].

Table 2.2: Operational parameters for detectors.

Parameter	Explanation
Duty cycle	Percentage of how much of a time period the system is active.
Exposure time	The time for which the detector collects light.
Frame rate	How often the sensor captures an image, expressed in FPS.
Pixel binning	Combining multiple pixel charges in both the horizontal and vertical direction to form a single larger charge or super pixel [27].

Pixel binning

Pixel binning, or just binning, is the process of combining multiple pixels in both the horizontal and vertical direction to form a single super pixel [27]. This super pixel represents the area of all the pixels which are contributing to the charge. A binning where the signal arises from a single pixel is denoted by 1×1, as seen in Figure 2.8. Similarly, a binning consisting of four adjacent pixels is denoted by 2×2 which increases the area and the sensitivity to light by a factor of four. Binning will lower the resolution of the image, but will decrease the amount of data to be saved.

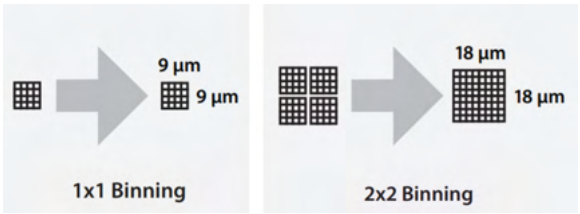


Figure 2.8: Illustration showing how the binning process combines adjacent pixels into a super pixel [27].

2.2.5 CMOS detectors for orbital detection

CCDs have previously dominated the field of scientific imaging due to many of the advantages they had over CMOS imaging technology [9]. However, development of the CMOS detector has lead to a decreased usage of CCDs. To capture the faint, small and fast-moving space debris objects it becomes important to use a detector with low readout noise and low dark noise [9]. Other important parameters are short read out times, high well capacity, high quantum efficiency and a low number of dark and hot pixels. With respect to these requirements, the CMOS offer a broad

range of advantages in comparison to CCDs due to their fast imaging and fast read out capabilities in combination with low noise characteristics.

2.2.6 Storing the captured image

In an image sensor the photon image on the detector is converted to a charge image in the pixels [16]. The (x, y) location of the charge in the image then mimics the location of the arriving photons. The readout process converts each charge packet to a voltage and digitize it in order to associate a number to each pixel location. These numbers are stored in a two-dimensional array such that the correct intensity is associated with the appropriate (x, y) pixel, this array is the digital image. There are various ways of storing the resulting images in electronic forms. Each of the many ways to organise the parameter and pixel data inside the file is called a format [30].

A common file format used for astronomical images is the Flexible Image Transport System (FITS) [17]. Various versions of FITS exist, but they all consist of a header, the image in binary form and an end section. The header can contain various information such as the number of bits representing the data in the image, the observed object, the telescope used, details of the observation, and other comments.

2.2.7 Determining detectability by Signal Noise Ratio

The ratio of signal amplitude to the root mean square (RMS) amplitude of the background fluctuation is called Signal Noise Ratio (SNR) [10]. SNR is a way of measuring the detectability of a signal. As mentioned in Section 2.2.3, the detector collects photons but also unwanted noise from various sources. The total noise is formed by noise from four independent noise sources: background noise, object noise, thermal signal, and readout noise.

The random fluctuation in a signal has a Poisson distribution and is referred to as *Poisson noise* or *shot noise* [10]. The Poisson distribution describes the random fluctuation in a signal with a constant average, i.e. the arrival rate of photons from a source. The probability $p(n, t)$ of n photons falling on a given area of a detector in a time t is given by

$$p(n, t) = (Nt)^n \frac{e^{-Nt}}{n} \tag{2.5}$$

where N is the average flux (photons per unit time)[10]. For the Poisson distribution, the RMS fluctuation in the average flux N is equal to $\sqrt{N}$. This means that the standard deviation σ_{SD} of the individual measurements from the true signal S can be estimated as

$$\sigma_{\mathrm{SD}} = \sqrt{S} \tag{2.6}$$

where S is the signal, and σ_{SD} the corresponding noise level [18]. This allows the total noise to be calculated as

$$\sigma_{\mathrm{SD}} = \sqrt{\sigma_{\mathrm{B}}^2 + \sigma_{\mathrm{S}}^2 + \sigma_{\mathrm{T}}^2 + \sigma_{\mathrm{R}}^2} \tag{2.7}$$

where σ_{SD} is the total noise, σ_{B} is the background noise, σ_{S} is the object noise, σ_{T} is the thermal signal, and σ_{R} is the readout noise. Using this noise equation, the SNR can be calculated as [18]

$$\mathrm{SNR} = \frac{S}{\sigma_{\mathrm{SD}}}. \tag{2.8}$$

By combining equations (2.6), (2.7), and (2.8) it is possible to calculate the SNR in one pixel in the detector as

$$\mathrm{SNR}_{\mathrm{pixel}} = \frac{S}{\sqrt{S + B + T + \sigma_{\mathrm{R}}^2}} \tag{2.9}$$

where S is the signal from the object collected in the pixel, B the signal from the sky background, T the thermal signal collected by the pixel, respectively, and σ_{R}^2 is the readout noise for one pixel [18]. This equation can be used to calculate the peak SNR from the brightest pixel in the image.

2.3 Visibility of objects

In order to visually observe a satellite it must be visible from the location on Earth and in sunlight. In this section the requirements to observe satellites and how to model their visibility are discussed, together with how the brightness of stars is measured.

2.3.1 Magnitudes

Apparent magnitude

The brightness of an object observed from Earth is measured by apparent magnitude m. Measuring m is called photometry. To calculate the magnitude of an object it is necessary to determine that magnitude 0 corresponds to some pre-selected flux density F_0 [Wm^{-2}], and all other magnitudes are then calculated as

$$m = -2.5 \log_{10} \frac{F}{F_0} \tag{2.10}$$

where F is the the observed flux density [11]. This equation implies that the brighter an object is, the lower its magnitude number is. By convention, at all wavelengths magnitude 0, and thereby F_0, has been attributed to the bright star Vega [10]. This implies that all objects brighter than Vega have negative magnitudes.

Equation (2.10) can be used to show that the relation between two objects of magnitude m_1 and m_2 is

$$m_1 - m_2 = -2.5 \log_{10} \frac{F_1}{F_2}. \tag{2.11}$$

A difference of 1.0 in magnitude corresponds to a star which is 2.512 times brighter.

The UBV and the Gaia G magnitude scales

Magnitude can be complicated since light is not monochromatic and therefore covers many wavelengths. Accurate photometry is accomplished with photoelectric and solid-state devices and filters that accept only certain wavelength bands [10]. The combined effect of the filter, any prisms, the transmission, the optical path, and the detector quantum efficiency, as a function of wavelength, defines the *passband* which needs to be calibrated for the specific instrument [31]. To account for these differences, many different photometric systems have emerged. It should be noted that they differ in central wavelength and bandwidth, which also depend on instrumental responses particular to each observatory [10]. A common photometric system is the UBV system which also covers bands in the red and infrared spectrum. When magnitude is given without specifying the band further, it is usually the normal magnitude in the V ("visual") band with the midpoint $\lambda = 550$ nm that is referred to.

It happens that passbands need to be re-calibrated for the same instruments during the lifetime of the instrument. This is true for the ESA space observatory Gaia which was launched on December 19 in 2013, and whose mission is to measure the positions, distances, motions and photometry of stars [32]. The mission data is made available to the public in different releases, and due to this cyclic processing of the Gaia data leading to the publication of better and more complete releases, the internal average instrument generated at each cycle is different [31]. Gaia has one passband per instrument; the white light G-band ($330 - 1050$ nm), the blue G_{BP}-band ($330 - 680$ nm) and the red G_{RP}-band ($640 - 1050$ nm) band [33].

Converting magnitudes to electrons

Star catalogues usually provides the magnitude of the star, but the detector measures photons and converts this to electrons. In order to understand what the detector actually measures it is necessary to be able to convert from a given magnitude to the resulting number of electrons in the detector. The flux of the star can be calculated by solving equation (2.10) for the flux F which results in the equation

$$F = F_0 10^{-m/2.5} \qquad [\mathrm{Wm}^{-2}] \tag{2.12}$$

where F_0, as previously mentioned, is the pre-selected total flux density corresponding to Vega which has been defined as the magnitude 0 reference star. Flux can be given in different units depending on whether if it refers to the flux at a certain

wavelength or to the total density. In equation (2.12) the flux F is the total flux density. Flux at a certain wavelength is denoted by F_λ [Wm^{-3}].

The energy of a single photon of wavelength λ equals hc/λ [J], where $h = 6.62607015 \times 10^{-34}$ [Jm] is Planck's constant and $c = 2.99792458 \times 10^8$ [m/s] is the speed of light. The number of electrons $S(\lambda)$ detected by a telescope with area A_{tel} in a wavelength interval $\Delta\lambda$, transmitted by an optical system of efficiency τ onto a detector of quantum efficiency Q_{eff}, during an exposure time t_{exp} can then be calculated by

$$S(\lambda) = \frac{\lambda}{hc}\tau Q_{\mathrm{eff}} A_{\mathrm{tel}} \Delta\lambda F_\lambda t_{\mathrm{exp}} \qquad \text{[electrons]} \tag{2.13}$$

where F_λ depends on the wavelength and can be calculated from equation (2.12) [16]. The efficiency τ is the product of all present transmission or reflectance factors in the system and will therefore depend on the used setup and on the observed wavelength.

2.3.2 Observer-geometry

Observing a satellite in daylight is difficult because the sky is very bright and the satellites are dim due to the reduced reflected solar flux at large *phase angles* [34]. The phase angle, ϕ, is the Sun-satellite-Earth angle whose vertex is at the satellite, and it is illustrated in Figure 2.9. The phase angle varies from 0 to 180°,

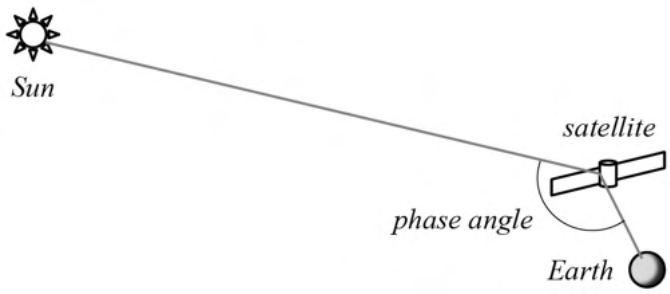


Figure 2.9: Phase angle geometry [34]. The shown geometry is for daytime.

and observations of satellites in GEO are routinely performed by optical sensors at night when the phase angle is less than about 85° [34].

In order for sunlight to reach the satellite it must be outside the Earth shadow. The main two regions of the shadow are called *umbra* and *penumbra*. Umbra is the central, completely dark part of the shadow, while penumbra is the portion of the shadow within which part of the disk of the Sun is still visible [10].

2.3.3 Albedo

The definition of the term *albedo*, denoted by a_0, is the ability of a body to reflect light [11]. Bodies in the solar system reflect sunlight, and the brightness of a body depends on its distance from the Sun and Earth, and on the albedo of its surface.

When doing optical observation of space debris the apparent brightness is measured [2]. If the observation geometry is known, the apparent magnitude may be converted to absolute magnitude, but this requires some assumption about the albedo which is most of the time unknown. The National Aeronautics and Space Administration (NASA) has conducted optical measurements in LEO to derive the albedo of known small-size debris, and first results indicated a mean albedo value between 0.1 and 0.2, an estimate which is based on assumptions about the shape and scattering properties of the debris [2]. When modelling the magnitude of satellites in geostationary orbit (GEO), albedo values from 0.175 to 0.2 have been used for the satellite bus [35].

The Earth albedo varies from about 0.02 to 0.8 depending on the region and angle of incidence, but it is common to use an average value of 0.3 [35]. This value is important when calculating the *eartshine*, which is the fraction of sunlight reflected from the Earth to the satellite location.

2.3.4 Reflecting area of an object

A large reflecting area makes a satellite easier to detect since it reflects more light. The reflecting area of a satellite is often approximated by its radar cross-section (RCS) which is denoted by σ. The RCS of a target in a radar beam is the projected area of a metal sphere that would scatter the same power in the same direction that the target does [36]. This is not true for Earth observers who see the reflected light on the satellite from the Sun, but it is a decent assumption in order to make an estimation about the magnitude of the satellite. RCS can be calculated by

$$\sigma = 4\pi R^2 \frac{P_s}{P_i} \qquad [\mathrm{m}^2] \tag{2.14}$$

where R is the distance to the target which is assumed to be far away, P_s is the scattered power density, and P_i is the incident power density at the scattering target [36]. P_s and P_i are measured in $\mathrm{W/m}^2$. The RCS depends on many different aspects and is a function of, for example, position of the transmitter and receiver relative to the target, the target geometry and material, wavelength, and polarisation.

2.3.5 Modelling apparent visual magnitude of a satellite

To model the photometric signature of a satellite it is common to assume the satellite is a sphere with a Lambertian surface [35]. A Lambertian surface is defined as an absolutely white, diffuse surface which reflects all radiation [11]. Using this

assumption, the phase function F_{diff} which describes the fraction of incident solar flux reflected by the spherical satellite as a function of the phase angle ϕ, can be calculated as

$$F_{\text{diff}}(a_0, r_{\text{sat}}, \phi) = a_0 \frac{2}{3} \frac{r_{\text{sat}}^2}{\pi R^2} (\sin\phi + (\pi - \phi)\cos\phi) \tag{2.15}$$

where a_0 is the satellite's albedo, r_{sat} is the radius of the satellite, R is the range between the satellite and the observer, and ϕ is the phase angle [35, 37]. By assuming that the optical cross section of the sphere is the same as the RCS, the satellite's radius can be calculated by

$$r_{\text{sat}} = \sqrt{\frac{\sigma}{\pi}} \tag{2.16}$$

where σ is the RCS given in square meters [35]. Equation (2.15) can then be written as

$$F_{\text{diff}}(a_0, R, \phi, \sigma) = a_0 \frac{2}{3} \frac{\sigma}{(\pi R)^2} (\sin\phi + (\pi - \phi)\cos\phi). \tag{2.17}$$

Using equation (2.11), the apparent visual magnitude of the Lambertian sphere is then calculated as

$$m_{\text{sat}}(\phi, R) = m_{\odot} - 2.5 \log_{10}(F_{\text{diff}}) \tag{2.18}$$

where the apparent visual magnitude of the sun $m_{\odot} = -26.74$ [35].

2.4 The night sky

A satellite is observed on the night sky. Different aspects of the night sky, such as for example different coordinate systems and the influence of the atmosphere, are discussed in this section.

2.4.1 Time

There are many different ways to define time, and there are several different time scales in use. Some of them, such as solar and sidereal times, are are based on the rotation of the Earth, while other scales such as dynamical and atomic time are based on different processes [11, 38]. In this section, some fundamental concepts which are related to time and used in astronomy are presented.

Coordinated Universal Time

The most commonly used time system is Coordinated Universal Time (UTC) which is derived from atomic clocks and which divides time into days, hours, minutes and seconds [38]. UTC makes use of *leap seconds* which are regulated by the U.S. Naval Observatory. UTC does not change for daylight saving time.

Epoch

The epoch is a particular instant of time which is used as reference in the determination or measurement of celestial object positions [10]. Positions are given for an epoch, and to obtain positions at some other epoch, the effects of proper motion, nutation, and stellar aberration must be included in the calculations.

Julian date

In astronomy it is common to use the Julian Date (JD), which is the number of days and fraction of days since noon on 1 January 4713 BCE [10]. Other epochs exists, such as the truncated JD which instead uses midnight on May 24, 1968 [39]. Currently, the most used epoch is noon of January 1, 2000 [11]. This is denoted as the J2000 epoch and represents the JD 2,451,545.0.

2.4.2 Coordinate systems

On Earth, the terrestrial parameters *latitude* and *longitude* are used to describe different locations. To locate celestial objects, there are a few different coordinate systems in use. In this subsection, the two most important systems are introduced: the equatorial coordinate system and the horizontal coordinate system. These coordinate systems are summarised in Table 2.3.

Table 2.3: Summary of the two most common coordinate systems [10].

Coordinate system	Reference plane	Reference direction	Latitude coordinate	Longitude coordinate
Equatorial	Celestial equator	Vernal equinox, γ	Declination, δ/DEC	Right Ascension, α/RA
Horizontal	Horizon	North	Altitude, h	Azimuth, A

Equatorial coordinate system

The equatorial coordinate, seen in Figure 2.10a, is the most common coordinate system in astronomy [39]. The system projects the equator of the Earth onto the celestial sphere to form the celestial equator, and uses this as reference plane. The reference direction is taken as the vernal equinox which is the point on the celestial sphere where the ecliptic (apparent path of the Sun on the sky) ascends and intersects the celestial equator [10].

The angular distance eastwards, along the celestial equator, between the object and the vernal equinox (denoted by γ in Figure 2.10a) is known as the object's right ascension and is denoted by α or RA [39]. RA is typically measured in hours,

minutes, and seconds. The angular separation between the celestial equator and the object is known as the object's declination and it is denoted by δ or DEC. Objects above the celestial equator have a positive declination, and objects below have a negative declination. The declination is measured in degrees, minutes, and seconds. However, both RA and DEC can be given in decimal degrees.

(a) Equatorial coordinate system [10]. **(b)** Horizontal coordinate system [40].

Figure 2.10: Two of the most common coordinate systems. Different angles and symbols are explained in the text.

Horizontal coordinate system

The horizontal coordinate system (also known as the AltAz system), seen in Figure 2.10b, is the simplest coordinate system [39]. The reference plane is the the tangent plane of the Earth passing through the observer [11]. This horizontal plane intersects the celestial sphere along the horizon. The point located straight above the observer is called zenith, while the point straight below the observer is called nadir. Circles that pass through these two points are called verticals and are perpendicular to the horizon.

Coordinates are given by altitude/elevation and azimuth. The altitude, h, is measured from the horizon along the vertical that passes through the object. The altitude ranges from -90° to 90° with negative angles for objects below the horizon, and positive values for objects above the horizon. The second coordinate azimuth, A, is the angular distance of the object's vertical from a fixed direction. This direction varies, but is usually either from the north or south. Direction is usually clockwise, but not always. Values lie in the range from 0° to 360°, or from -180° to 180°. In Figure 2.10b, the azimuth is measured eastward from north.

2.4.3 The atmosphere

Ground-based telescopes observe light coming through the atmosphere which causes many problems. The foremost problems are that it absorbs light from targets,

and scatters light into the telescope from non-targets [17]. Phenomena in the atmosphere such as rain, clouds and snow are also highly inconvenient for observations. The atmosphere therefore has a major impact on image quality due to turbulence [16]. The atmosphere disturbs the incoming waves which limits the ability of the telescope to reach its ultimate angular resolution. Transmission and absorption also happens in the atmosphere, and different wavelengths are absorbed differently. The number of electrons $S(\lambda)$, presented in equation (2.13), is therefore further reduced by a transmission factor which depends on wavelength and varies from 0 to 1.

Turbulence and inhomogeneity in the atmosphere also cause *scintillation* and *seeing* [17]. Scintillation is seen as the twinkling of the stars due to the rapid change in the brightness of a stellar image when more or less light from the incoming point source is scattered by irregularities in the atmosphere. Seeing is when the stellar image slightly moves away from its true position due to refraction at boundaries between different layers within the atmosphere. Both of these phenomena change with time and images obtained with exposures longer than around 0.01 s will therefore be blurred into the *seeing disk* which contributes to the PSF. Some of the ways to reduce these phenomena are to use adaptive optics or to observe at high altitudes, or even in space.

2.4.4 Sky glow and the sky background

The night sky is never completely dark, and on the ground at visible and near-infrared wavelengths the atmosphere contributes light from several sources: *airglow*, scattered sunlight, starlight, and moonlight, and scattered artificial light [19]. Light scattered by dust and molecules causes the sky background to have a certain intrinsic brightness [17]. This phenomena imposes a limit upon the faintest object that is detectable through the telescope. The main source of scattered light is artificial light, specifically street lighting. Observational sites are therefore preferably located far away from urban areas. Apart from artificial light sources, the Moon also has a major impact on the night sky which varies with the lunar cycle.

Airglow is caused by light that is emitted by excited atmospheric molecules [11]. However, most of the radiation is in the infrared domain and therefore less problematic for optical observations.

2.4.5 Star catalogues

The first star catalogue was published in the second century, but today there exist several catalogues which to varied extensions cover different magnitudes, positions and properties [11]. Some of the most important ones in use today have been summarised in Table 2.4. Catalogues can broadly be divided into stellar and nonstellar, but some are mixed [39]. These catalogues contain subdivisions based on whether data is observed visually, photographically or digitally. Another parameter of star

catalogues is their *completeness* which is the number of detected objects compared to the number of expected detected objects from current computer models.

The first astrometric satellite Hipparcos was launched in 1989 by ESA [11]. The satellite measured exact positions of more than a hundred thousand stars and resulted in the Hipparcos catalogue which contains astrometric and photometric data of 118,000 stars down to a precision of milliarcseconds. Observations were made in the B and V bands, with a limiting magnitude of 12.4 in the V band [39]. The satellite measurements also resulted in data for the Tycho-1 and Tycho-2 catalogues which are less precise than the Hipparcos catalogue, but contain data for about one million respectively 2.5 million stars. These cover stars down to limiting V magnitude of about 11.5.

Another catalogue is the U.S. Naval Observatory USNO-B1.0 Catalog which contains data for 1,024,618,261 stars and galaxies down to magnitude 21 [11, 39]. The data is based on images from several photographic sky surveys and consists of right ascension and declination, proper motion, and magnitude estimates.

In 2013, the satellite Gaia was launched by ESA [11]. The satellite is a successor to the Hipparcos satellite and has improved the accuracy to about 10^{-5} seconds of arc, while also providing an order of magnitude more objects in more bands and to fainter magnitudes [11, 39]. Gaia has photometric uncertainties in the millimagnitude range and positional uncertainties in the range of hundredths of milliarcseconds [39]. The magnitudes range from 3 to 21 in the G band, and the current latest data release Gaia Early Data Release 3 (GEDR3) contains 1,811,709,771 sources and is essentially complete for G magnitudes between 12 and 17 [41]. This makes it the most extensive star catalogue so far.

Multiple catalogues can be composed to a larger database to be cross-matched with relative ease. The most important online database is the Set of Identifications, Measurements, and Bibliography for Astronomical Data (SIMBAD), which is formed from the concatenation of a large number of astronomical catalogues [39]. SIMBAD contains data for about 5,800,000 stars, and additionally data about 5,500,000 nonstellar objects [42].

Table 2.4: Summary of different star catalogues used today.

Catalogue name	Limiting magnitude	Number of objects	Comment
Gaia EDR3	3 - 21	1,811,709,771	
Hipparcos	12.4	118,000	
SIMBAD	-	5,800,000	Concatenation
Tycho-2	11.5	2,500,000	
USNO-B1.0	21	1,024,618,261	Includes galaxies

2.4.6 Available astronomy software tools

There exists quite many different astronomical observatory softwares used to display the night sky at a given time. Some prominent examples are Stellarium [43], Cartes du Ciel [44], and Google Sky [45]. A similar tool is HeavensAbove which provides detailed star charts where trajectories of passing satellites are shown [46]. Python also has a collection of software packages called `Astropy` which contains many different functions applicable to astronomy [47]. One noteworthy package is `astroquery` which is used for querying online databases such as the Gaia catalogue [48]. Another Python package useful for modelling telescope optics is `Poppy`, which includes a system for modelling a complete optical instrument, including optical propagation and PSFs [49].

ESA is currently working on a project called Pyxel which is a general detector simulation framework [50]. The goal is to simulate a variety of imaging detector effects combined on images made by CCD or CMOS-based detectors. The tool is still under development, but a beta version is available.

2.5 Orbits

In order to observe a specific satellite it is critical to understand its orbit. This section describes the classical orbital parameters, how satellites' orbital parameters are distributed, and how to propagate an orbit from known data.

2.5.1 Keplerian elements

In order to define the state of a satellite in space, six quantities are required [38]. The classical orbital elements are the Keplerian elements, which are illustrated in Figure 2.11.

The first two Keplerian elements describe the shape and size of the ellipse in which the satellite travels:

1. Semimajor axis (a) - the sum of the distance between the extreme points of the orbit (periapsis and apoapsis) divided by two.

2. Eccentricity (e) - describes how elliptical the orbit is. If $e = 0$ the orbit is a perfect circle.

The second two parameters defines the orbital plane of the ellipse:

3. Inclination (i) - describes the tilt of the orbital plane from the reference plane and ranges from 0° to 180°.

4. Longitude of the ascending node (Ω) - The angle in the reference plane from the reference direction to the ascending node, it varies from 0° to 360°. For geocentric orbits where the reference plane is the equatorial plane, and the

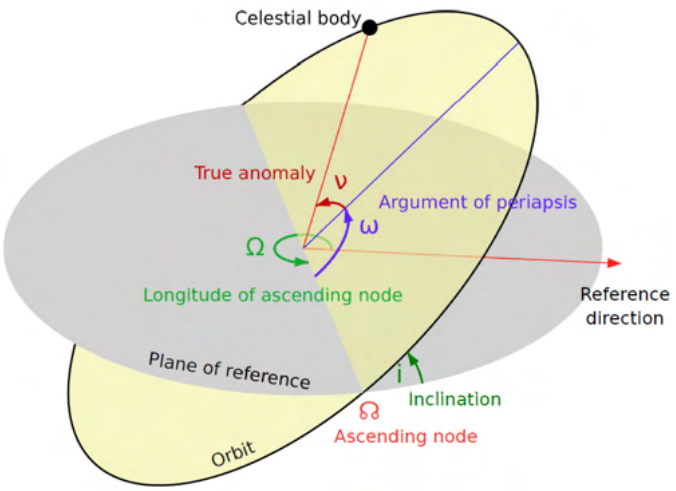


Figure 2.11: Orbital parameters [51].

vernal equinox is used as reference direction, the longitude of the ascending node is called the right ascension of the ascending node (RAAN).

The last two parameters are:

5. Argument of periapsis (ω) - the angle measured from the ascending node in the direction of the satellite's motion to the closest point of the orbit (the periapsis). It varies from 0° to 360°.

6. True anomaly (ν) - varies from 0° to 360° and determines the satellite's position relative to the location of periapsis at some specific epoch.

2.5.2 Two-Line Element Sets

The classical orbital elements are widely used, but it is also common to use a Two-line element set (TLE) to describe the satellite's orbit at a certain epoch [38]. It is based on UTC and consists of two lines which are presented in Tables 2.5 and 2.6, the set may additionally have a line 0 containing the common name for the object based on information from the satellite catalogue [52]. TLEs are publicly available through the websites `www.space-track.org` and `www.celestrak.com/NORAD/elements/`.

The satellite catalogue number is assigned to the object by the US Space Force and is assigned sequentially as objects are catalogued [52]. The mean anomaly (line 2, field 7) is not a Keplerian element but is a way of indicating where the satellite would be if it was moving in a circular orbit of radius a [11]. The mean

motion (line 2, field 8) is the mean angular velocity given in revolutions per day, and it is often used to replace the semimajor axis a using Kepler's laws [38].

TLEs are only valid for a short time and rapidly go out of date [53]. It is therefore important to pay attention to the epoch of the TLE which informs about the date on which the TLE is the most accurate. Typically, TLEs elements are only useful for about one to two weeks before and after the epoch. Historical TLEs can be requested from `www.celestrak.com/NORAD/archives/request.php` for example. The maximum accuracy of a TLE is limited by the number of decimal places in each field, and in general TLE data is accurate to about a kilometer or so at epoch before quickly degrading [54].

Table 2.5: The TLE elements in the first line [52].

Line 1			
Field	**Columns**	**Element**	**Example**
1	1	Line number	1
2	3-7	Satellite catalogue number	25544
3	8	Classification (U=Unclassified, C=Classified, S=Secret)	U
4	10-11	International Designator (last two digits of launch year)	98
5	12-14	International Designator (launch number of the year)	067
6	15-17	International Designator (piece of the launch)	A
7	19-20	Epoch Year (last two digits of year)	08
8	21-32	Epoch (day of the year and fractional portion of the day)	264.51782
9	34-43	First Derivative of Mean Motion (Ballistic Coefficient)	.00020137
10	45-52	Second Derivative of Mean Motion (decimal point assumed)	00000-0
11	54-61	Drag Term (decimal point assumed)	16538-3
12	63	Element Set Type (always zero in distributed TLE data)	0
13	65-68	Element set number. Incremented when a new TLE is generated.	292
14	69	Checksum (modulo 10)	7

Table 2.6: The TLE elements in the second line [52].

Line 2			
Field	**Columns**	**Element**	**Example**
1	1	Line Number	2
2	3-7	Satellite catalogue number	25544
3	9-16	Inclination (degrees)	51.6335
4	18-25	RAAN (degrees)	344.7760
5	27-33	Eccentricity (decimal point assumed)	0007976
6	35-42	Argument of Perigee (degrees)	126.2523
7	44-51	Mean Anomaly (degrees)	325.9359
8	53-63	Mean Motion (revolutions per day)	15.7212
9	64-68	Revolution Number at Epoch (revolutions)	32890
10	69	Checksum (modulo 10)	6

2.5.3 Orbit propagation

Orbit propagation is done to examine how the effect of perturbations on the satellite will affect the orbital elements [38]. General perturbation techniques replace the original equations of motion with an analytical approximation that captures the essential character of the motion. This approximation is valid over some limited time interval. A table of predicted positions of a body in the solar system or of a spacecraft is called an *ephemeris* [10].

The most classical approach to orbital propagation is to use Keplerian motion and two-body equations which provide exact solution to the equations of motion for two mutually attracting bodies [38]. However, these equations only consider the force of gravity and no other perturbations. They will therefore not provide reliable results for satellites over time.

In order to take perturbations such as Earth oblateness, lunar and solar gravitational effects, gravitational resonance effects and drag into account, the Simplified General Perturbations (SGP4) was developed to propagate TLEs for near-Earth satellites [38, 55]. Near-Earth objects were defined as space objects with a period less than 225 minutes, and objects with a period greater than or equal to 225 minutes were defined as deep-space objects [55]. Later, the SGP8 model was developed to handle deficiencies of SGP4 for the special cases of orbital decay and reentry [38]. These models have later been improved by various contributors. The most recent update was by Vallado et al. who released an updated and combined set of code, test cases, results and analysis for the SGP4 routine in 2006 [54].

There are many versions of the basic satellite prediction algorithms, and perfect agreement between two different prediction softwares should not be expected [53]. The satellite's orbit should be expected to constantly change as the SGP4 propagation routine models effects like atmospheric drag and the Moon's gravity. This is

particularly true for the true anomaly parameter which can swing wildly for satellites with nearly circular orbits since the satellite's perigee can be moved by even slight perturbations to the orbit.

2.5.4 Available software tools for working with orbits

TLEs are distributed by websites such as Space-Track [56] and CelesTrak [57], which continuously update their TLEs. These websites can also provide additional information about the satellites, i.e. an estimated RCS.

The Python package `Skyfield` [58, 59] provides positions for the stars, planets, and satellites in orbit around the Earth, together with a large range of different functions such as orbital propagation. Similar functions are provided by the low level space dynamics library `Orekit`, which aims at providing accurate and efficient low level components for the development of flight dynamics applications [60]. The library is written in Java, but a Python wrapper is available. Another software tool which is used by many space organisations is Systems Tool Kit (STK) [61]. STK includes extensive functions for orbit determination and visualisation.

Chapter 3

Development of the simulation tool

This chapter describes the process of developing the simulator, the different methods applied, and assumptions made in the process. It starts with defining the requirements for the simulator tool which were to be fulfilled, and continues by presenting the final system architecture of the tool. Then the process of implementing the requirements is discussed.

3.1 The requirement matrix

The first step in developing the tool was to decide on the requirements to be fulfilled by the finished simulation tool. The requirements were prioritised and divided into hard and soft requirements. The hard requirements are identified by the word "shall" which indicates that they must be implemented. The soft requirements are identified by the word "should" and indicates they should be implemented if times allow once the hard requirements are fulfilled. Each requirement also has a verification method showing how to verify whether the requirement is fulfilled or not. Some of the requirements also have a rationale where such is relevant.

The full concatenated requirement matrix is seen in Appendix A in Table A.2, together with the requirement matrix legend in Table A.1. The following subsections present the different subgroups of requirements and the reasons behind them. The word *system* is used for the combination of a specific telescope and a specific camera. The word *camera* is synonymous to the word *sensor* and refers to the CMOS sensor chosen for the simulation.

3.1.1 Requirement group 1: Basic properties

The first group of requirements presents the basic properties of the tool, and they are seen in Table 3.1. These requirements are the most fundamental and represent some of the basic objectives of the simulation tool.

Table 3.1: Requirement group 1: Basic properties requirements

ID	Requirement	Verification Method	Rationale
B-1	The code shall be written in Python.	Inspection	
B-2	The tool shall be able to simulate stars down to the visual magnitude limit of the telescope parameters.	Analysis	
B-3	The code shall be able to plot a satellite tracklet.	Inspection	Objective
B-4	The system parameters shall include camera and telescope properties.	Test	Objective
B-5	The tool should have a graphical user interface.	Inspection	Facilitate user interaction

Requirement B-1 is a hard requirement given by SSC, and one of the main reasons behind it is to facilitate the use of the many preexisting libraries written for Python. Python code is also very readable which facilitates for other developers to read and modify the code. Besides that, Python is easy to run on different operating systems which makes it very portable.

Requirements B-3 to B-4 are also hard requirements and represent basic objectives of the tool in order to be able to use it for its intended purpose: investigate feasible observational scenarios. To be able to see how the stars are affected by different disturbances it must be possible to know which stars are observable in perfect conditions. This is the reason behind requirement B-2. Requirement B-3 is also a fundamental objective. It should be noted that when using satellite tracking this requirement would be applicable to star tracklets. Requirement B-4 is also a fundamental objective. Different observational systems have different abilities depending on the combination of camera and telescope. It is therefore necessary to be able to simulate the images as they would look like from different systems.

Requirement B-5 is listed as a soft requirement since the tool does not require a graphical user interface (GUI) to run, but that it would be beneficial to have in order to facilitate the user interaction with the simulation tool.

3.1.2 Requirement group 2: Input parameters

The second requirement group is the required input parameters to the tool, and they are listed in Table 3.2. These are necessary in order to fulfil the basic properties and other requirements. All the input parameters requirements were identified as hard requirements necessary to fulfil the tool purpose.

Requirements I-1 and I-2 are related to the location where the observation is taking place, which is chosen by the user. In order to be able to simulate the stars

Table 3.2: Requirement group 2: Input parameters requirements

ID	Requirement	Verification Method	Rationale
I-1	The tool shall be able to recieve geographic coordinates as input.	Test	In order to test different observation sites
I-2	The tool shall be able to recieve observation altitude as input.	Test	
I-3	The tool shall be able to recieve time as input.	Test	
I-4	The user shall be able to choose between sidereal or satellite tracking.	Test	
I-5	The tool shall be able to recieve exposure time as input.	Test	Will affect length of tracklet.
I-6	The tool shall be able to import satellite orbital information.	Test	
I-7	The tool shall be able to recieve system parameters.	Test	Sensor size, pixel size, noise figure.
I-8	The tool shall be able to recieve binning as an input.	Test	In order to test different resolutions.
I-9	The tool shall be able to recieve image format as input.	Test	

and satellite from a given location it is necessary to know the coordinates of it. The altitude of the location is also important to have an idea about the achievable seeing at the location. The next requirement, I-3, represents the starting time of the observation. It is fundamental to understand the position of the satellite relative to the observer and which stars that are visible during the observation. Requirement I-4 was chosen in order to be able to switch between satellite and sidereal tracking by the telescope since both were feasible scenarios according to SSC. Requirement I-5 represents for how long the telescope should observe, and this parameter will affect both the length of the tracklets but also how many photons the telescope have time to gather from the targets.

To understand how the satellite tracklet will look like it is necessary to have some orbital information about the satellite, and this was the reason behind requirement I-6. The idea was that the tool should be able to import and read a text file containing the orbital information in form of a TLE. The following requirement, I-7, is related to requirement B-4 which needs this input in order to be fulfilled. Requirement I-8 allows the user to choose to use binning, and requirement I-9 lets the user choose how to save the resulting image from the simulated observation.

3.1.3 Requirement group 3: System properties

Requirement group three contains the requirements related to the system properties, and they are presented in Table 3.3. Requirement P-1 is a hard requirement and reinforces that the resulting image should take parameters specific to the chosen

camera into account. Some examples of these parameters are sensor size, pixel size, read noise and dark current. Requirement P-2 is also a hard requirement and very similar to P-1 but for the telescope. Some examples of telescope parameters to apply when modelling the image is the aperture of the telescope, its focal length, any mirror obstruction, and the efficiency of the telescope for the observed wavelength.

Requirement P-3 is a soft requirement since this is a more complicated process and not fundamental to be able to use the simulator for its purpose.

Table 3.3: Requirement group 3: System properties requirements

ID	Requirement	Verification Method	Rationale
P-1	The tool shall model the image based on camera properties.	Test	Sensor size, pixel size, noise figure.
P-2	The tool shall model the image based on telescope properties.	Test	Aperture, focal length, obstruction, mirror/lense type and properties.
P-3	The tool should model lens deformation of the telescope.	Analysis	

3.1.4 Requirement group 4: Disturbances

The fourth requirement group consists of the disturbances requirements. Requirement D-1 is a hard requirement and deals with the thermal noise. Requirements D-2 to D-3 are the disturbances that should be modelled if possible. These include the atmospheric effects from winds and clouds, mount disturbances, the effect from moon and sky glow, and dead pixels (both dark and hot). These are soft requirements since some of them are more complicated to model, but also because the final images will still be useful even without these disturbances.

Table 3.4: Requirement group 4: Disturbances requirements

ID	Requirement	Verification Method	Rationale
D-1	The tool shall model thermal noise.	Analysis	
D-2	The tool should model atmospheric effects.	Analysis	Winds, jet streams, upmoving air, transparency.
D-3	The tool should model mount disturbances.	Analysis	
D-4	The tool should model the effect of moon glow.	Analysis	
D-5	The tool should model the effect of sky glow.	Analysis	
D-6	The tool should model dead pixels.	Test	

3.1.5 Requirement group 5: Satellite observation properties

The last group of requirements contains the requirements related to satellite observation properties. These are all soft requirements since some of these properties are hard to find information about and predict. It was also believed that modelling of them might be more complicated compared to other requirements. Summarising these requirement will influence how visible the satellite is from the observation location, meaning they will affect the apparent magnitude of the satellite.

Table 3.5: Requirement group 5: Satellite observation properties requirements

ID	Requirement	Verification Method	Rationale
S-1	The tool should take the albedo of the satellite into account.	Test	
S-2	The tool should take the area of the satellite into account.	Test	
S-3	The tool should take the observation geometry into account, i.e. sun angles.	Analysis	Will affect shadowing and satellite visibility

Requirement S-1 states that the tool should take the albedo into account, and a high albedo will result in more reflected sunlight from the satellite. The following requirement, S-2, is also related to the reflected sunlight since a large area may be able to reflect more light from the sun towards the observer. The last requirement, S-3, states that the observation geometry should be used in the modelling. This refers to the geometry between the observation location on Earth, the satellite location, and the sun which will determine whether the satellite is sunlit or not.

3.2 System Architecture

The second step of the development of the simulation tool was to define an initial system architecture. This was done in order to start planning how the tool would work, what type of code that was necessary to write, identify which packages that might be useful, and to have a map of which functions the tool would consist of. The system architecture was expanded and reworked during the coding phase, and the final version of it is seen in Figure 3.1. The system architecture also serves as an introduction to the tool before going into detail about the implementation of the different functions.

The functions have been numbered in order to facilitate the reading of it. The simulation begins with all the inputs being read and handled to the tool core, this has been marked with the number one to indicate this is the first process. The processes then occur counterclockwise, with the second process being responsible for handling information about the observing location to the observation location function, which returns the observation location object.

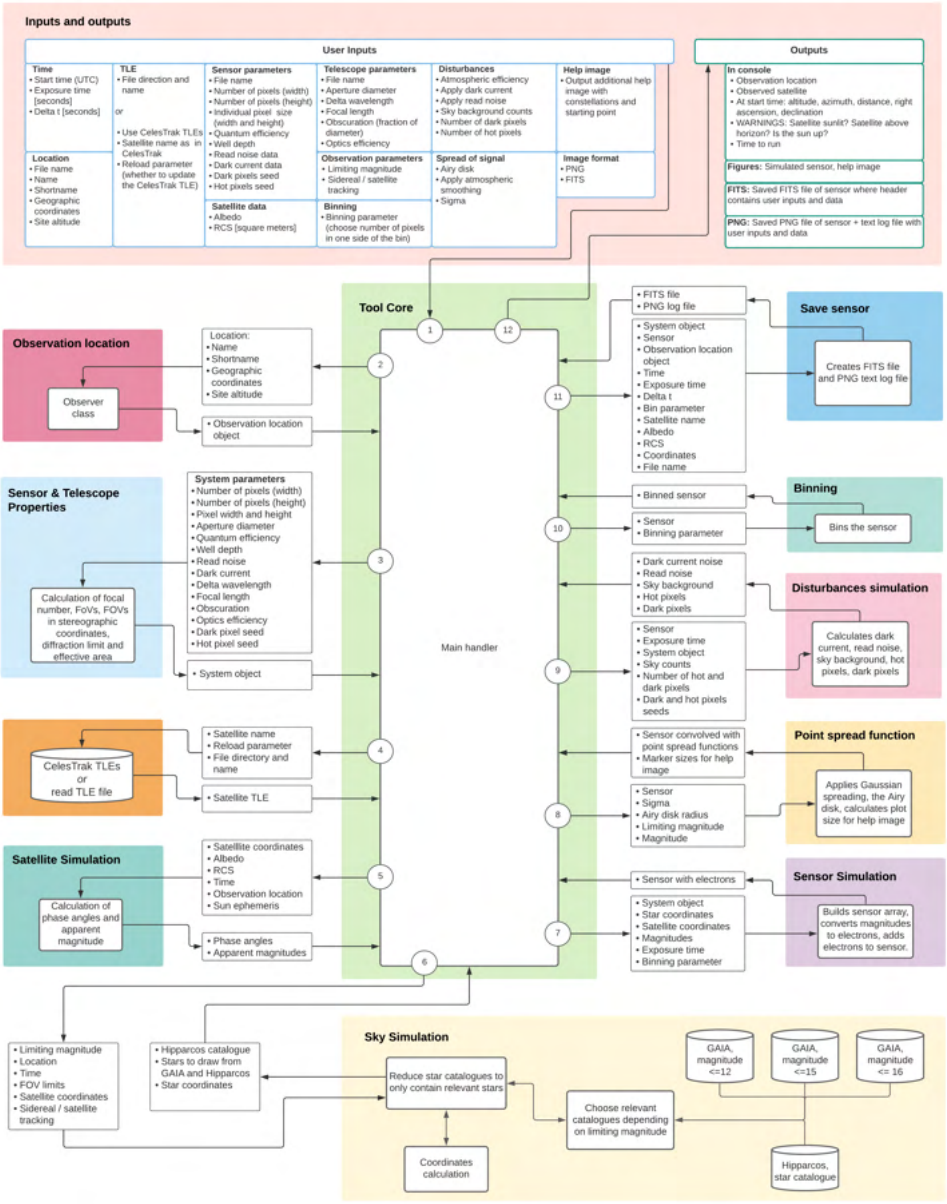


Figure 3.1: System architecture of the simulation tool. The processes happens in the numbered order, with inputs delivered to the tool core being the first process, and displaying the outputs being the last process.

The processes then continue sequentially in the numbered order. The last process is outputting the simulated sensor and additional information. The simulated sensor is also saved as a FITS file and/or a Portable Network Graphics (PNG) file depending on what the user inputted. The different functions, their inputs and outputs, and their Python module dependencies are explained in the following section.

3.3 Implementation of the tool functions

This section explains the decisions and assumptions made when developing the functions for the simulation tool. It also explains what inputs are required to the functions, what they output, and which modules and libraries the functions depend on.

3.3.1 Initial modelling

The simulation tool can be divided into three parts: modelling the sky correctly, modelling the satellite tracklet, and modelling the optics of the observation system. Proper modelling of the sky is an essential part to work in order to be able to evaluate the other two parts, and this was therefore the chosen starting point. When searching for a way to do this, the Python package `Skyfield` [58, 59] was encountered. The package website included some example code and corresponding plots, out of which one depicted the position of the Comet NEOWISE during a few days in July 2020 [62]. This plot is seen in Figure 3.2. The code merged four different types of data sources: a planetary ephemeris, a comet orbit database, a large star catalogue, and constellation diagrams. The code was also able to correctly depict an object in orbit onto the star background, while taking the FOV of the observer into account.

After some investigation, `Skyfield` was selected because of its many useful functions that was expected to benefit the code development. The package deploys functions to deal with time and positions, it could easily calculate sunsets and sunrises for different locations, check whether a satellite was sunlit, import ephemeris files, import the Hipparcos star catalogue, and it could also work with TLEs. Additionally, the package was open source and the code was uploaded to GitHub.

Skyfield generates a barycentric position measured from the gravitational center of the solar system. All vectors therefore originate at the gravitational center of the solar system. The package also includes the function to ask for an astrometric position relative to a specific observation location, and this position is adjusted for light-time delay. This astrometric position can also be given in the equatorial or the horizontal coordinate system.

The ephemeris files for the positions of the Sun, planets and their moons are provided by NASA's Jet Propulsion Laboratory (JPL) which offers high accuracy

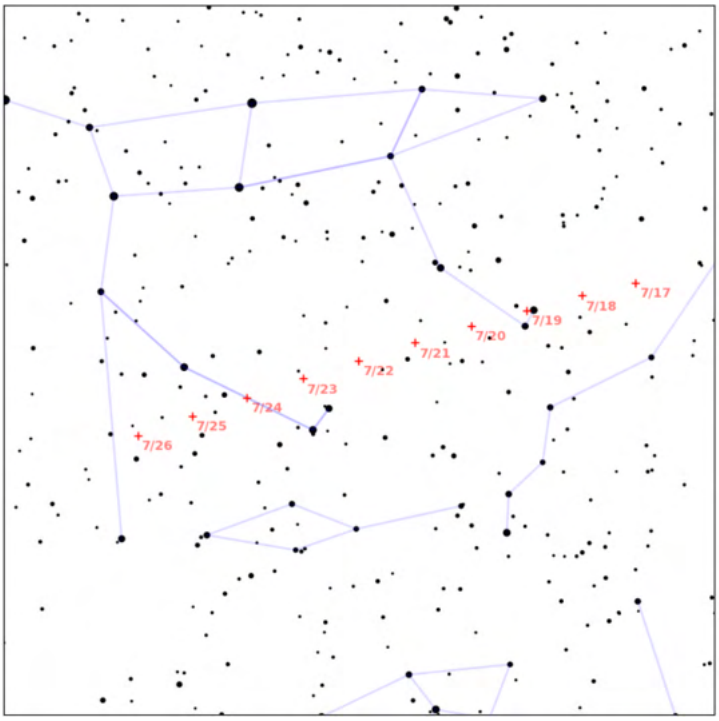


Figure 3.2: Example plot generated by Python package `Skyfield` [62]. The plot shows the course of the Comet NEOWISE across the sky from July 17 to July 26, 2020.

tables of positions for time spans ranging from decades to centuries [63]. After looking through the different options from JPL available through `Skyfield` the short ephemeris de440s issued in 2020 was chosen for the simulator. The file was small enough to be able to handle easily, included effects not taken into regard in the long files, and was also the latest released which means it is believed to be the most accurate ephemeris.

`Skyfield` also includes functions related to projections, and to build the chart seen in Figure 3.2 a stereographic projection was used. When projecting a sphere onto a plane, it is impossible to conserve both area, shapes and angles simultaneously. The stereographic projection is a projection method that preserves angles, and it has long been used for making maps [64].

Prediction of Earth satellites by TLEs is also possible to do using `Skyfield` which runs them through the SGP4 satellite propagation routine [53]. The module documentation about Earth satellites offered an example of how to load a TLE for the International Space Station (ISS) from CelesTrak and propagate it. By implementing this example and merging it with the example used to generate the

plot in Figure 3.2, changing the colours of the sky and the constellations lines, changing the marker of the satellite trajectory, removing the dates, and adding a marker to indicate the start position of the satellite tracklet, the plot seen in Figure 3.3 was produced. This plot shows the trajectory of ISS as seen from Stockholm, with a FOV of $5^\circ \times 5^\circ$, with an exposure time of 3 seconds. The size of the star markers depend on the magnitude of the star, and the chosen limiting magnitude.

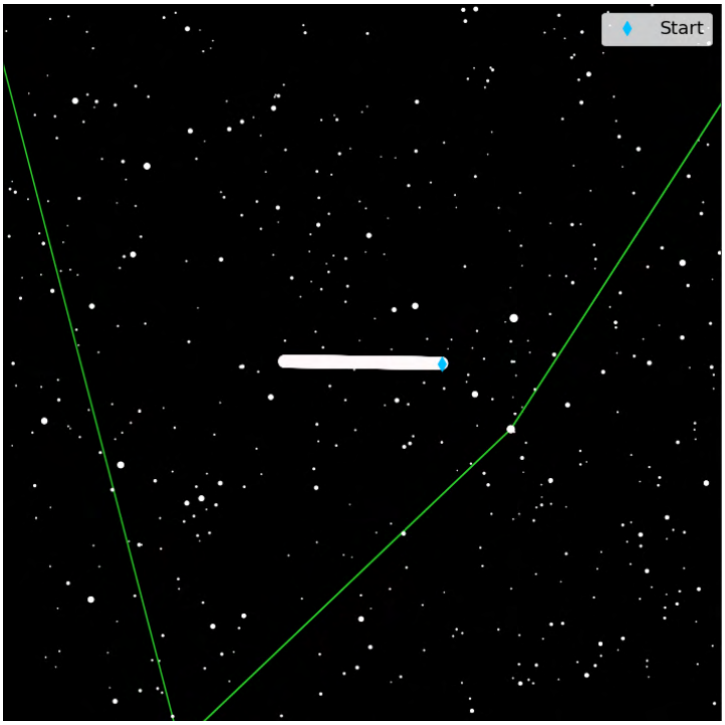


Figure 3.3: An initial plot showing the trajectory of ISS as seen from Stockholm. The exposure time is 3 seconds, and the FOV $5^\circ \times 5^\circ$.

3.3.2 Initial decisions

To proceed, some fundamental decisions regarding design and the simulator limitations were taken. These decisions are explained in this subsection and their respective outcomes are discussed.

Time

An early question about the inputs to the simulator regarded time. Working with different time zones can be confusing, and the concept of daylight saving can cause

further confusion. It was therefore concluded that the best way would be to only work with times in UTC. This is an important point in terms of the inputs to the simulation tool; the starting time must be given in UTC. It is up to the user to input the corresponding time in UTC if the satellite pass is given in local time, for example by checking passes using `www.heavens-above.com`.

Two other inputs also refers to time: the exposure time, and the Δt. The exposure time is straight forward, this is how long the simulated telescope is gathering photons, and the exposure time should always be given in seconds. The Δt is the time step and determines how often the different positions of the satellite should be calculated and plotted. Having a too small Δt will cause the tracklet to appear as individual dots, instead of as a coherent tracklet. The effect of varying the Δt is further discussed in Section 3.3.5.

Observation location object

The second process in the system architecture is where the data about the observation location is sent to the observer class, which returns an observation location object. This object has five attributes: name, short name, latitude, longitude, and elevation in meters above sea level (MSL). A function to the class which prints the name of the current observation location is also available. An example of an observation location object's attributes is seen in Table 3.6.

Table 3.6: Example of the attributes of an observation location object.

Name	Acronym	Latitude	Longitude	Elevation
Stockholm	Sthlm	59.33 N	18.07 E	28 MSL

The observation class was the first class written for the simulator. It is useful to create objects in order to decrease the amount of variables that needs to be inputted to different scripts and functions.

The observation location data is inputted through a text file which is read by the simulator. When using the tool, it is important to provide the information in the correct format. This format is illustrated in Section 4.1.1, Figure 4.2. It is especially important that the data is given on the fifth line in the presented order, and that the different data points are separated by a comma and a space.

The long name is used when saving information about the simulated sensor, but the short name is used in the title on the plot that is outputted from the simulator in order to save some space when having long names. The latitude and longitudes are inputted as numbers, and the elevation of the location is inputted in meters. The elevation parameter is used to build the stereographic projection and to calculate the distance to the satellite. The altitude information could be used in the future to calculate the expected seeing conditions at the location, which is a consideration for future iterations of the code.

Centering the image on the satellite

When using sidereal tracking the satellite positions are calculated for each time step, but the star positions are only calculated once. The time for the stars calculation was chosen as the middle time of the satellite times. This time is later used when calculating the stereographical projection and means that the resulting image will be centered on the middle of the satellite tracklet.

When instead using satellite tracking, the same range of satellite times is calculated but only the first point of time is plotted, see Section 3.3.5 for details. This initial point of time is used to calculate initial star positions, and this means that the image will be centered around this point. The satellite position will therefore be in the middle of the image.

Excluding the planets, the Moon, and Earth

The positions of the planets are not calculated or shown in the simulator images. The positions can be calculated easily, but the their appearance in the image is more complicated and this work was therefore de-prioritised and is left as future work.

A similar approach is used to neglect the implementation of the Moon. The effects and size of the moon depends on when in the lunar cycle the images are simulated, and therefore implicates an even more complicated modelling process which was de-prioritised in favor of developing more important functions. However, the Moon will also contribute to the amount of sky glow. A parameter to adjust the amount of sky background noise has been implemented and is discussed further in Section 3.3.10. This parameter can be increased if the observer knows that the Moon will present during the observation, but the function does not include the possibility to simulate a gradient sky background in this iteration.

To observe a satellite in real life it needs to be positioned above the horizon, otherwise the Earth is blocking the view. The tool gives a warning if the altitude of the satellite is less than 0° during the simulated time period. This warning is further discussed in Section 3.3.6. Additionally, observing a satellite close to the horizon is not ideal due to the large amount of air between the satellite and the observer which causes disturbances. A function to simulate atmospheric smoothing has been implemented and can be increased if the user is aware of the satellite being close to the horizon. The details of this function is discussed in Section 3.3.10.

3.3.3 Choosing star catalogues

As discussed in Section 2.4.5, there are several star catalogues in use in the field of astronomy. Choosing the catalogue to be used for the simulator therefore required some considerations. The result is a combination of two catalogues: Hipparcos and GAIA EDR3. This choice and the result of it is motivated in this section, together with how the catalogues are implemented.

Combining the Hipparcos and GAIA EDR3 catalogues

Since the Hipparcos catalogue was used for the initial example this was easy to implement in the simulation tool. However, the Hipparcos catalogue only includes stars to magnitude 12.4, and this was deemed as insufficient since observations are planned to take place at locations with very good seeing conditions where much fainter stars should be visible. To accommodate for this, the GAIA EDR3 catalogue was implemented in the simulation tool. This catalogue was chosen due it being the most extensive and its data being the latest. The catalogue could also be queried easily by using Astronomical Data Query Language (ADQL) which is implemented in the `astroquery` module.

The GAIA EDR3 catalogue has a lower limiting magnitude of 3, and the final star catalogue used in the simulator consists of both Hipparcos stars and stars from GAIA EDR3 since they complement each other. This means that there is an overlap between the catalogues for the bright stars. Ideally, this should not cause much problems since the coordinates should be similar enough to plot the star in the same position, but this has not been investigated extensively. The GAIA EDR3 catalogue is incomplete for magnitudes below 12, and the method to investigate the exact overlap between the catalogues is not straight forward. Practically, this means that some bright stars might appear as a double star or as a brighter star than it should. However, this should be case for quite few stars and the investigation of this is left as future work. Using the Hipparcos catalogue is also necessary to be able to draw the constellation lines which were seen in green in Figure 3.3.

Choosing the GAIA EDR3 magnitude passband and its consequences

In Section 2.3.1 the different passbands of the Gaia instruments were discussed. In order to use these magnitudes it was necessary to choose one of the passbands. Evaluation of one of the specifications of one of the sensors to be used by SSC concluded that the wavelengths with $Q_{\text{eff}} \geqslant 50\%$ were in the range from $\sim$425 nm to $\sim$775 nm. The decision was therefore to use the G-band (330$-$1050 nm) magnitudes since these sensor wavelengths were a subset of the wavelength range of the G-band. It is important to notice that this choice implicates that the observational wavelength is hard coded in the simulation tool to 700 nm, which is roughly the middle of the GAIA passband. The width of the passband can be chosen by the user as an input, but the center of the passband can not. This is a limitation of the tool which was deemed reasonable for the scope of the project. This limitation could be re-evaluated in the future for a new iteration of the code.

Accessing GAIA EDR3

Accessing the GAIA EDR3 catalogue is, as mentioned previously, done by writing queries in ADQL. Queries include the catalogue name, the data categories to be fetched, and other limitations. The catalogue can also be queried for a certain position given in right ascension and declination, with a given radius. However,

running these queries are very time consuming and Gaia users are limited to a run time of 120 minutes. The whole catalogue is also of considerable size, the complete catalogue is 616 GB [65]. Queering the catalogue for each simulation size was therefore not an option due to the long run times. Instead, the catalogue has been queried for varying magnitude ranges and later merged into complete catalogues. The data columns included in these catalogues are: source id, right ascension [degrees], declination [degrees], magnitude and epoch year. Additionally, a column with right ascension in hours has been calculated and added to accommodate the information `Skyfield` requires for the stereographical projection.

In total, data for 157,645,865 stars with a magnitude less than or equal to 17 has been fetched. These have been divided into three catalogues. The sizes and coverage of the catalogues, together with the Hipparcos catalogue, are presented in Table 3.7.

Table 3.7: Magnitude ranges and sizes for the star catalogues used in the simulation tool.

Catalogue name	Magnitude range	Number of stars	Size
stars_HIP	$\leqslant 14.08$	118,218	8.11 MB
GAIA_maglim_12	$\leqslant 12$	3,087,828	129 MB
GAIA_maglim_15	$\leqslant 15$	36,908,086	1.51 GB
GAIA_maglim_16	$\leqslant 16$	77,936,605	3.19 GB

The choice of dividing the stars into different catalogues is related to run time. The number of stars that needs to be evaluated is one of the main contributors to an increased run time. Running the simulator using sidereal tracking, with a limiting magnitude of for example 10, takes roughly half a minute, obviously depending on other parameters and the capabilities of the computer used to process the code. Increasing the limiting magnitude to, for example, 15.5 increases the run time by another 5 minutes, leaving all other parameters equal. Notice that even though additional data for magnitudes between 16 and 17 has been fetched, this has not yet been merged into a single catalogue. This data consists of another 79,709,260 stars of size 3.27 GB, and results in a run time which is too long for currently acceptable limits, and may also cause the memory to run out. The limit of magnitude 16 has been deemed acceptable, it is nonetheless straightforward to create and implement a more extensive catalogue.

Working with a large database

The resulting catalogues have been saved as *pickle files* [66] which are stored in the same directory as the code. When the simulation tool runs, it unpickles the relevant catalogue depending on the limiting magnitude specified by the user. The tool makes a first reduction of the data by removing all stars from the catalogue

which is less than or equal to the limiting magnitude. Coordinates for each star is then calculated from the stereographic projection, and only the stars within the limits set by the FOV are kept. The file containing all the original data is then deleted from the memory in order to not risk a memory overflow.

3.3.4 Retrieving, importing, and propagating TLEs

TLEs are read and imported in two ways by the simulator: from a text file, or directly from CelesTrak. The user specifies which way to input the TLE by a True / False parameter. If the TLE is to be read from a file the user needs to specify further the local url to the file directory, and the name of the text file. The text file should contain the name of the satellite in the first line, then the first TLE line, and lastly the second TLE line. An example of this format is seen in Figure 3.4. TLEs can be fetched and requested from, for example, `www.celestrak.com/NORAD/elements/`, or from `www.space-track.org`.

```
STARLINK-1794
1 46695U 20073AB  21073.94031125  .00001507  00000-0  12004-3 0  9993
2 46695  53.0555 150.2827 0001140  96.4633 263.6486 15.06405770 22988
```

Figure 3.4: Example of input format when the user specifies the TLE in a file.

If the TLE is to be read directly from CelesTrak, the user needs to specify the name of the satellite exactly spelled as in `www.celestrak.com/NORAD/elements/active.txt`. The satellites in this website are the currently active satellites, and also indicates which satellites the user can choose from. The TLEs are updated continuously, and the user can choose whether to reload the latest version or to use the already fetched file in the directory. If the TLE file is to be reloaded, the progress of downloading it is indicated by a progress bar in the console.

It was also investigated whether the simulation tool should be able to run a query directly to `Space-Track` to fetch TLEs, and what this implementation would look like. The investigation deemed this possible, but the implementation of it was de-prioritised in favour of more urgent tasks since requirement I-6 was already fulfilled by the existing import modes. The benefit of implementing this function is, for example, that `Space-Track` can provide an indication of the RCS of the satellite directly, instead of requiring the user to input the RCS in square meters.

`Skyfield` used the SPG4 propagation routine to predict the positions of Earth satellites from TLEs [53]. The package has implemented the corrected and updated version of the algorithm that was released in 2006 [54].

3.3.5 Implementation of satellite tracking

Requirement I-4 stated that the user should be able to choose between sidereal and satellite tracking, and the difference between these modes was illustrated in

Figure 1.2. This requirement has has been implemented by a True / False parameter, where True means satellite tracking, and False indicates sidereal tracking. The implementation of satellite tracking is derived from the satellite tracklet calculated when using sidereal tracking. If the telescope was to track a satellite along this trajectory, the star tracklets would have the same tilt as the satellite tracklet, but in the opposite direction. This means, that for each coordinate pair in the satellite tracklet, a corresponding pair needs to be calculated for each star visible in the FOV. Only the first coordinate pair is then plotted for the satellite to indicate the tracking of it.

The first step in the simulator when calculating the corresponding star coordinates is to know which stars that will be visible in the final image. For sidereal tracking this was simply done by checking which stars are in the FOV, but when tracking the satellite stars outside the original FOV might enter the image during the exposure. This is done by calculating the length of the satellite tracklet by using the Pythagorean Theorem. This length is the maximal distance travelled by the satellite during the exposure, and by adding this distance to the FOV limits when reducing the star catalogue it is possible to know which stars might enter the image.

The second step for calculating the star coordinates is to to off-set the origin. In the original $(x, y)-$coordinates calculated by the stereographic projection, the origin is located in the middle of the image. This might cause complications, and to simplify the calculations all coordinates are offset by the FOV-limits. This causes the origin to be located in the lower left corner of the image and all coordinates are then located in the first quadrant and therefore positive.

Calculating the additional star coordinates is then carried out by looping over the satellite coordinates. For each coordinate pair in the satellite tracklet, a Δx_{sat} and Δy_{sat} is calculated. A new star coordinate pair $(x_{j,\text{star}}, y_{j,\text{star}})$ is then calculated for each star by

$$x_{j,\text{star}} = x_{i,\text{star}} - \Delta x_{\text{sat}} \tag{3.1}$$

$$y_{j,\text{star}} = y_{i,\text{star}} - \Delta y_{\text{sat}} \tag{3.2}$$

where $(x_{i,\text{star}}, y_{i,\text{star}})$ is the previous coordinate pair. An example of the resulting image when using satellite tracking is seen in Figure 3.5.

The run time when using satellite tracking is heavily influenced by the number of coordinates that are required to be calculated. This is influenced by the distance travelled by the satellite, the exposure time, the chosen Δt, the FOV, and the limiting magnitude. If the satellite is observed closer to the horizon, the tracklet looks shorter compared to an observation closer to zenith relative to the observer. This is because the satellite is much further away from the observer when being close to the horizon. A longer exposure time will also result in a longer tracklet. A longer tracklet requires a smaller Δt parameter in order to look like a smooth tracklet, instead of individual dots, and the number of coordinates that are calculated for the satellite equals the exposure time divided by Δt. The number of stars that require

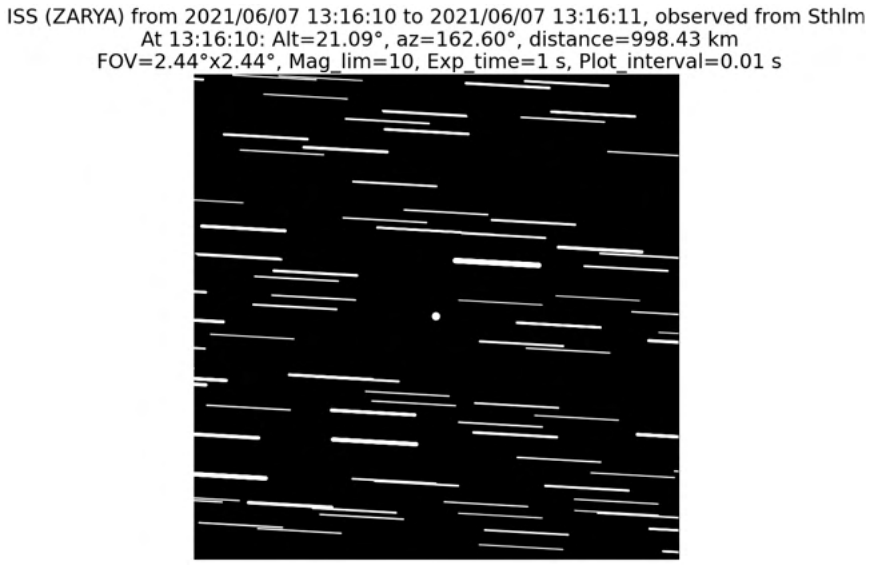


Figure 3.5: Example of the implementation of satellite tracking.

calculation of additional coordinate pairs depends on how long the satellite tracklet is since more stars can enter into the image, but also on the limiting magnitude which can drastically increase the number of stars visible in the FOV. A larger FOV will also include more stars than a small FOV. All combined, these parameters can cause the run time to vary from seconds to several minutes. A recommendation to the user would therefore be to start simulating with a lower limiting magnitude and larger Δt to have an idea what the image will look like. Once the user is pleased with the result, the parameters can be adjusted to allow for a more realistic observation scenario.

3.3.6 Visibility of the satellite

Up to this point in the development of the simulation tool, the user could simply input the magnitude of the satellite manually. In order to accommodate requirements S-1 to S-3, a more sophisticated way of calculating the apparent magnitude of the satellite was required.

Implementing the satellite apparent magnitude equations

Calculating the apparent magnitude of a satellite was discussed in detail in Section 2.3.5. The user inputs the albedo and an estimated RCS in square meters, the distance to the satellite is calculated by `Skyfield`, and the required missing parameter to implement equations (2.17) and (2.18) is then the phase angle ϕ. As

explained in Section 2.3.2, and illustrated in Figure 2.9, the phase angle is the angle between the Sun and the observer as seen from the satellite. `Skyfield` includes a function which returns the angle between two different positions [67], and this was used to calculate the phase angle for each time step. Implementing equations (2.17) and (2.18) was then straightforward. The result is a calculated apparent magnitude of the satellite for each time step.

It is important to note that the calculated apparent magnitude should only be taken as a rough estimate. The real apparent magnitude also depends on the orientation and tumbling of the satellite, for example. Since the Earth shadow has been neglected from the modelling, the apparent magnitude will be incorrect for the time periods the satellite spends in eclipse. The result is also very influenced by the choice of albedo and RCS, and it can be difficult to find accurate values for these parameters.

Warnings related to satellite visibility

Observation of a satellite usually requires that the satellite is sunlit, that the sky at the observation location is dark enough, and that the satellite is above the horizon. The sky background parameter can be increased to simulate a brighter sky, but these factors are otherwise not included in the modelling of the tracklet image. However, the simulation tool does warn the user if one of these conditions is not fulfilled during the observation period.

`Skyfield` has a function to calculate whether an object is sunlit or not, and this function returns a list of True / False corresponding to each time step [68]. The tool goes through this list, and if a False is found the tool prints, for example, "WARNING: At 2021/06/09 21:56:30 UTC the satellite is in shadow" where the time is the corresponding time step where the False was found.

A similar approach is used to check whether the sun is up or not at the observation location by using one of `Skyfield`'s built-in functions [69]. This function uses the official definition of sunrise and sunset from the United States Naval Observatory, which defines them as the moment when the center of the sun is 0.8333 degrees below the horizon. Using this function another list of True / False is generated where True corresponds to that the sun is up. The simulation tool goes through this list, and if a True is found it prints the warning "WARNING: At 2021/06/09 08:20:30 UTC the sun is up".

To check whether the satellite is above the horizon, the tool goes through the altitude coordinates of the satellite during the observation period. The altitude coordinate is relative to the observer, and altitude = 0° is defined as the horizontal plane. The tool therefore looks for any negative altitude coordinates, and if one is found it prints "WARNING: At 2021/06/09 08:20:30 UTC the satellite is below the horizon". It should be noted that in reality observations close to the horizon are not performed due to the amount of disturbances by the atmosphere. If another value than 0° is better suited to the user, this warning is easy to change.

For a more accurate modelling it would be beneficial for the project to consider switching to implement functions from the `Orekit` package, which can provide trigger warnings for these types of events.

3.3.7 Modelling the system properties

Up to this point in the tool development, the FOV to use for the simulated observations was directly inputted by the user. However, in reality the FOV depends on the combination of sensor and telescope as was presented in Section 2.1.2. A first step to implement the system properties was therefore to calculate the FOV the system would result in. This was done by implementing equation (2.4), which requires the user to input the number of pixels in each side of the sensor, the size of the pixel, and the diameter of the telescope. The equation also requires the focal number N which is calculated by dividing the focal length by the aperture diameter. The focal length is also provided by the the user, and the simulation tool then calculates the focal number before calculating the FOV. The FOV will consist of two angles, one for each side of the sensor. The FOV is later re-calculated as a limit in stereographic coordinates to be able to use as a delimiter for reducing the star catalogue.

The diffraction limit is also a system property which depends on the observed wavelength and the telescope diameter, as was seen in equation (2.1). The tool calculates this parameter using the wavelength hard coded into the tool, 700 nm, and the aperture diameter specified by the user.

Similarly as for the observation location, a class was written to have the system parameters as an object with different attributes. The attributes are:

- Sensor width and height in number of pixels
- Individual pixel width and height in meter
- Sensor well depth
- Approximate quantum efficiency for 700 nm
- Sensor read noise data
- Sensor dark current data
- Dark and hot pixel seeds
- Telescope diameter
- Telescope focal number
- Telescope effective area
- Wavelength

- Passband width
- Diffraction limit
- FOV in degrees

The dark and hot pixels seeds are explained in Section 3.3.10. The effective area is the telescope area used to collect photons. This is calculated by the tool based on the aperture diameter provided by the user, and the obscuration by the secondary mirror which is inputted by the user as a fraction of the aperture diameter. Examples of these inputs are seen in Section 4.1.1.

3.3.8 Simulating the CMOS sensor

At this stage of the development, the modelling of the star and satellite tracklets were working well. However, the optics of the system had still not been considered much and the images were not very similar to real images taken from a professional telescope. After some discussion, it was found that the best way to tackle these issues and the modelling of the disturbances would be to move away from the plotting strategy and on to simulating the CMOS sensor itself. The idea was that this would allow easier implementation of the remaining system parameters, and that it would make it easier to model the disturbances.

The initial plot has, however, been kept as a possible output called *helper image* in the simulation tool. If the user chooses to output the helper image, the corresponding plot of the sensor is shown. The helper image contains the constellation lines and starting point of the satellite tracklet. This has been showed to sometimes be helpful to orient the image and compare the result with other sources, such as the plots produced by the website `www.heavens-above.com` for example.

Transforming magnitudes to number of electrons

The simulated sensor is modelled as a matrix by the simulation tool, and the number of columns and rows of the matrix correspond to the number of pixels in each side of the sensor. Each element of the matrix therefore corresponds to the value of a pixel in the sensor. After long discussions about what these values would represent, it was decided that each simulated pixel would have a value that corresponds to the number of electrons the pixel contains before the read out process. When the matrix is created it only contains zeroes which corresponds to that no electrons have yet accumulated in the sensor.

Each star and the satellite has an apparent magnitude, and to be able to know how many electrons each pixel will have these magnitudes needed to be transformed into electrons. Searching for a method to do this resulted in equation (2.13)

which was presented in Section 2.3.1. Using $\lambda = 700$ nm, this equation has been implemented as

$$S = \frac{700 \times 10^{-9}}{hc} A_{\text{effective}} \Delta\lambda F_\lambda t_{\text{exp}} \tau_{\text{system}} \qquad \text{[electrons]} \tag{3.3}$$

where h is Planck's constant, c the speed of light, $A_{\text{effective}}$ is the calculated effective area, and $\Delta\lambda$ and t_{exp} has been chosen by the user. The efficiency parameter τ_{system} represents the efficiency of the whole system and is calculated as

$$\tau_{\text{system}} = Q_{\text{eff}} \times \tau_{\text{optics}} \times \tau_{\text{atmosphere}} \tag{3.4}$$

where Q_{eff} is the quantum efficiency for the detector at $\lambda = 700$ nm, as inputted by the user. The parameter τ_{optics} is the optical transmission efficiency for the whole optical system and depends on the simulated telescope. It is also inputted by the user and should be a product of the transmission of the mirror, transmission by the coating of the lenses, and the internal transmittance of the lenses. The parameter $\tau_{\text{atmosphere}}$ is inputted by the user as well, and is meant to represent a general transmission by the atmosphere. This depends partly on the wavelength, but lowering this efficiency can be a way of simulating clouds for example.

Equation (3.3) still requires the wavelength dependent flux F_λ to be inputted. Studying the literature it was found that the absolute flux $F_{\lambda 0}$ at $\lambda = 700$ nm, corresponds to $F_{\lambda 0} = 1.76 \times 10^{-12}$ [W cm^{-2}µm^{-1}] [16]. This value can be converted to $F_{\lambda 0} = 1.76 \times 10^{-2}$ [W m^{-3}], and inputting this into equation (2.12) the flux corresponding to each magnitude can be calculated as

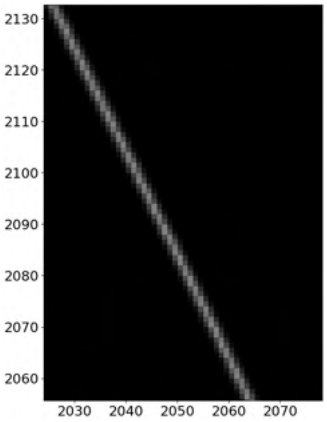
$$F_\lambda = 1.76 \times 10^{-2} \times 10^{-m/2.5} \qquad \text{[W m}^{-3}\text{]} \tag{3.5}$$

where m is the apparent magnitude of the star or the satellite. Inserting this into equation (3.3), each magnitude can be converted to a number of electrons which gather in the sensor pixels during the exposure time.

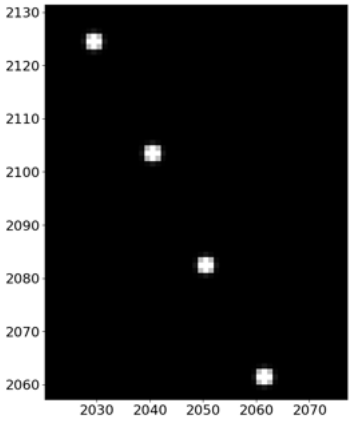
Calculating electrons per time step

The calculated electrons need to be added to the simulated sensor. The first step of this process is to calculate the number of electrons captured by the sensor in each time step. The total number of electrons which has been calculated for each star and satellite is for the whole exposure time, but if the object has a tracklet appearance this total number of electrons needs to be divided by the number of time steps to simulate the electrons formed during each step in the movement. The satellite electrons and star electrons are added to the sensor in two separate functions in the sensor simulation, but each function starts with checking whether satellite or sidereal tracking is used for the current simulation scenario. If satellite tracking is used, all star electrons are divided by the number of time steps used in the simulation. If sidereal tracking is used, the satellite electrons are divided by the number of time steps instead.

The division of electrons may cause errors when the Δt is large, and therefore less time steps are used. The simulation assumes the observation is continuous during the whole exposure time, but if too few time steps are used the object will look as discrete points instead of like a continuous tracklet, the difference is illustrated in Figure 3.6. It is therefore important to find a fine enough Δt depending on the observational scenario and system parameters, and to use this when generating the images that are wished to be kept.



(a) $\Delta t = 0.001$ s, resulting in a continuous tracklet.



(b) $\Delta t = 0.1$ s, resulting in a discrete tracklet.

Figure 3.6: Example of a continuous versus a discrete tracklet. These examples were generated using an exposure time of two seconds, leaving all other parameters unchanged except the Δt. The axis numbering is the pixel indices, and the colours have been scaled to the number of electrons in the pixels. It can be seen that the continuous tracklet is much fainter compared to the discrete points.

Adding electrons to the sensor

The process of adding electrons to the sensor utilizes the stereographic projection coordinates. As explained previously, the stereographic projection of the celestial sphere and satellite resulted in a set of (x, y)-coordinates which were offset to be strictly positive. These coordinates are transformed to match the pixel indices by calculating the width and height of the pixels in the stereographic coordinates, and then dividing the (x, y)-coordinates by these numbers. This results in that each coordinate pair is matched into a corresponding pixel position in the sensor.

As discussed in Section 3.3.5, a larger limit than simply the FOV was used to reduce the star catalogue when using satellite tracking to account for the fact that some stars might enter the image as the satellite is moving. This means that some of the newly calculated pixel indices might be out of range of the sensor size,

indicating they should not be visible in the resulting image. Each index is therefore checked if it is in the range of the number of pixels in the sensor before moving further. If this check results in that the electron number corresponding to the pixel index should be added to the sensor, the pixel value is updated to equal the sum of its previous number of electrons and the new number of electrons.

In a first iteration of development, this step included a check of whether the new pixel value would exceed the pixel well depth or not. If the new value exceeded the well depth, the pixel value was simply updated to equal the well depth instead to indicate saturation of the pixel. This was later changed, and the current iteration of the code checks the well depth of the sensor in a later step. The reasons behind this decisions are explained in Section 3.3.10.

3.3.9 Modelling the seeing and point spread function

In reality the sensor electrons would spread to more than one pixel due to the atmosphere and the diffraction by the telescope. In the original plot this had been approximated by calculating the marker size of the star and satellites by using the chosen limiting magnitude. This quite naive approach worked decently for most cases for the plot, but a better strategy is required for the simulated sensor.

The photon spread process due to the atmosphere and telescope is a complex problem, and a lot of effort has been put into investigating how to model this process in the simulation tool. This is due to the modelling of seeing being dependent on many factors such as the exposure time and system parameters for example. Different types of telescopes are either more limited by the diffraction of the telescope, or the disturbances by the atmosphere, depending on their specifications and the observational conditions. The resulting PSF from the telescope depends on many parameters such as the aperture for example, but also on the obscuration and placement of the secondary mirror if such is used. This section goes through the modelling process and motivates the current modelling used in the simulation tool.

Initial modelling of the point spread function

When first examining modelling the PSF of the telescope, the `Astropy` package `Photutils` [70] was considered. However, `Photutils` is based upon building an effective PSF from images containing already spread light sources, and then reconstructing the PSF function. This did not apply to the simulation tool and `Photutils` was therefore abandoned in favour of examining the `Astropy` package `Poppy` [49]. Initial examples from the `Poppy` documentation describes how the PSF is built up by having the user define a sensor and telescope, resulting in a PSF with effects from the aperture type and telescope obstruction. It is also possible to include diffraction spikes from the vanes holding the secondary mirror [71].

The method used by `Poppy`, and later also in the simulation tool, is based on finding a matrix called the *kernel* which represents the PSF. The simulated

sensor is then convolved with the kernel to apply the PSF spread of the electrons. `Astropy` includes two functions for convolution where one is implemented as a direct convolution algorithm [72], and the other uses a fast Fourier transform (FFT) [73]. According to `Astropy`, direct convolution is better for small kernels, while the FFT is much more efficient for larger kernels [74].

The implementation of `Poppy` was unsuccessful. The resulting kernel from using the parameters of the telescope as input to the function is illustrated in Figure 3.7. Applying this kernel to the sensor caused the original centered satellite tracklet to multiple into four separate tracklets located in each corner. This was deemed unacceptable and another approach was therefore examined.

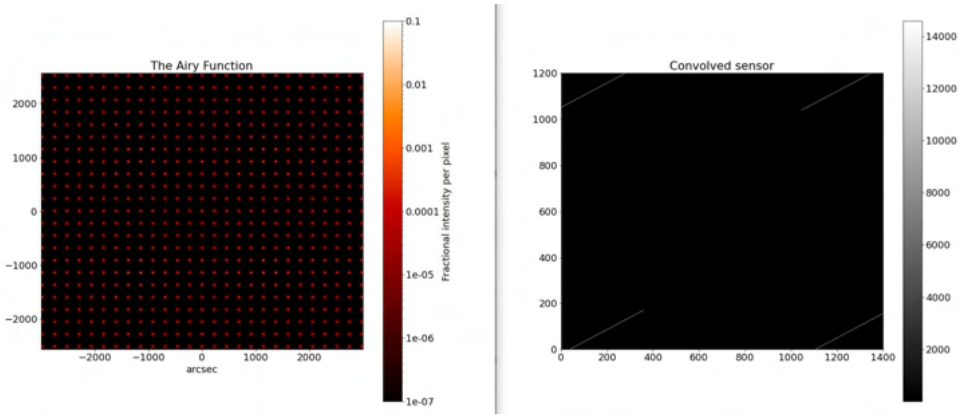


Figure 3.7: Example of the generated Airy disk by the `Astropy` package `Poppy`. The telescope inputs lead to the Airy disk becoming a grid, which resulted in the convolved sensor displaying four tracklets located in the corners instead of one centered tracklet.

Modelling the point spread function with `Astropy` kernels

The method used for implementing the PSF is based on the `Astropy` packet which includes a function to generate an Airy disk kernel in 2D [75]. The function takes a few different arguments, where the mandatory argument is the radius of the Airy disk kernel, which is measured from the centre of the Airy disk to the the first dark ring. When reading the documentation of the kernel function it was not obvious what this radius should be or what unit it should be given in. Literature stated, as mentioned in Section 2.1.2, that the radius to the first dark ring is equal to the diffraction limit θ. This limit was calculated in equation (2.1), where the unit is radians. It is also common to calculate this limit in arcseconds. Both these options were considered for the input, but the example given in the kernel documentation seemed to imply that the radius should be given in number of pixels instead. This

was implemented by first calculating the number of radians per pixel by dividing the FOV by the number of pixels in the sensor side. The radius in pixels is then calculated by dividing the diffraction limit in radians by the number of radians per pixel. This final number is used as input to the kernel function. The simulation tool then calculates the kernel matrix, and performs the convolution between the PSF function and the sensor by using FFT convolution. The FFT method was chosen since it is not dependent on the dimensions of the kernel, and since large kernel matrices might be used for the simulations. The result of the PSF convolution is that the pixel electrons are spread out to nearby pixels, where the spread depends on the diffraction limit of the telescope.

The Airy disk kernel function does not include modelling of diffraction spikes, and therefore the current iteration of the code does not include this phenomena. This was deemed acceptable for the scope of the project, but this could be of interest to revisit in the future since diffraction spikes can be difficult to handle by the software extracting satellite orbital data from the tracklets.

Using the Airy disk kernel is a simplified version of the PSF the telescope gives rise to. In reality, the secondary mirror in a reflector telescope and the vanes holding it would heavily impact the Airy disk pattern. However, these effects are not too visible on dim objects and the current iteration is still valuable for modelling observations with short exposure times or objects at relatively low magnitude.

Modelling the seeing with `Astropy` kernels

Next step is to model the spread and smoothing of the signal by the atmosphere. One way to model this phenomena is by applying a second convolution to the sensor with a Gaussian function. Similarly as for the Airy disk, `Astropy` includes a function to build a Gaussian 2D kernel [76]. The function takes several arguments, where the mandatory argument is the standard deviation of the Gaussian function, in number of pixels. In the simulation tool this parameter has been implemented as a user input called *sigma* after the standard notation of the standard deviation. The user chooses sigma, and a larger sigma models a more turbulent atmosphere with worse seeing compared to a case with a smaller sigma.

The Gaussian kernel function uses the sigma parameter to calculate the kernel matrix, and again convolution by FFT is used to apply it to the simulated sensor. The result is that the star and satellite electrons are further spread to nearby pixels. Since, in reality, the spread by the atmosphere takes place before the photons enter the telescope, this process has been placed before convolution with the Airy disk function in the simulation tool. Performing two convolutions can be quite time consuming, so if further work on the simulation tool would focus on optimising the tool performance this process could be re-evaluated.

Kernel sizes

Both the Gaussian and Airy disk `Astropy` functions take an optional argument which defines the size of the kernel matrix in the x- and y-directions. If no arguments are given, the default option is to use a kernel which has the number of rows and columns that corresponds to eight times the input parameter (sigma or radius in number of pixels). The size of the kernel affects how far the electrons can spread. For most stars this is not a problem, but for sources that give raise to many electrons, such as the satellite when satellite tracking is used, the electrons appear to "saturate" the kernel. The result is that the satellite appears as a square that does not blend smoothly with the background. This has been illustrated in Figure 3.8, which shows what the satellite looks like after the Airy disk and Gaussian smoothing with sigma = 1 have been applied. The figure has been created by opening the resulting FITS file in `ESA/ESO/NASA FITS Liberator`, zooming in on the satellite, and adjusting the black and white levels to enhance the saturation effect.

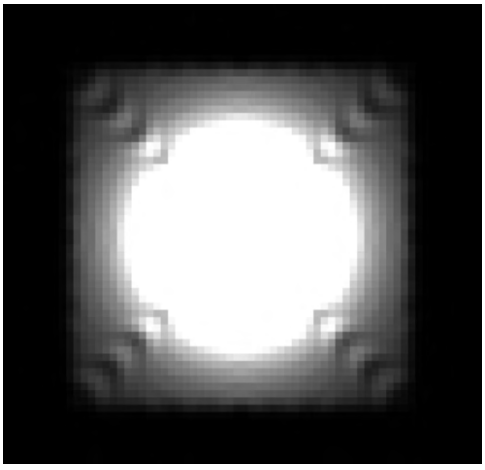


Figure 3.8: Example of the saturation effect which depends on the kernel size. This figure shows the simulated electrons from a satellite, spread by the Airy disk and Gaussian smoothing. In this simulation satellite tracking was used, and the apparent magnitude of the satellite was calculated to 0.014.

In the specific simulation used to generate Figure 3.8, the apparent magnitude of the satellite was calculated to 0.014. The size of the Gaussian kernel matrix was left at default, but the size of the Airy disk kernel has been increased to 100×radius, which in this example is equal to 46. The artefacts seen in each corner of the square is a product of using a too large kernel for the Airy disk. Using a large kernel might result in boundary effects since the convolution process needs to handle the boundary cases. How to handle the boundaries can be specified by an

argument to the FFT convolution function, and in the simulation tool the argument has been set to *wrap* which indicates a periodic boundary [73].

It is still rather unclear what exactly a "good" kernel size is since the effects depend on how many electrons the pixel to be spread contains, and the different system parameters such as the size of the sensor. The kernel sizes might need to be adjusted for different simulated systems, but in the current version of the simulation tool the Gaussian kernel size has been left at default, and the Airy disk size has been set to 50×radius. It might be that the most realistic kernel size should equal the sensor size. In the way the current iteration of the simulation tool, this size would result in boundary artefacts. One way to mitigate this issue could be to simulate a much larger image than what the telescope actually sees, perform the convolutions, and then only keep the actual part of the image seen by the telescope. However, this would increase the run time. Investigating this method further has been left as potential future work.

The Moffat kernel

As seen in the previous discussion, uncertainties still remain for the process of modelling the PSF and seeing. One possible alternative would be to use the Moffat kernel to model the seeing. The Moffat distribution models better the wings of the function than the Gaussian function. This especially applies for images dominated by seeing [19].

`Astropy` does include a kernel function for the Moffat distribution which requires two arguments called *gamma* and *alpha* [77]. These parameters describe the shape of the Moffat distribution, and it was suggested to find a suitable value for alpha and hard code it in the simulator, and then allow the user to choose gamma similarly to the strategy used for sigma when implementing the Gaussian kernel. However, finding a good value for alpha requires more investigation, and the effects of changing gamma needs to be examined further. Investing the implementation of the Moffat distribution kernel has therefore been left as potential future work as well, and the current version of the code applies the Airy disk and Gaussian kernels. The functions to use the Moffat kernel and also the functions from `Poppy` have been left in the simulation tool and can be called upon if these functions would be desired in future versions of the simulation tool.

3.3.10 Modelling the disturbances

The sensor has now been filled with electrons spread over several pixels. However, there are additional disturbances that will effect the final image. This section explains how the additional disturbances have been implemented in the simulation tool. Some of these implementations were inspired by the examples found in the *CCD Data Reduction Guide* [78].

Dark current

As explained in Section 2.2.3, the dark current is the current which is measured when no light is falling on the detector due to the photons generated by heat. Dark current is temperature dependent, and the user therefore needs to provide dark current data for the sensor which corresponds to the expected operational temperature. This number is usually found in the sensor data sheet in the unit [electrons/pixel/second], at a specific temperature.

The dark current increases linearly with the exposure time, and the simulation tool therefore calculates the total current by multiplying the dark current with the exposure time which results in the number of dark current electrons per pixel. This type of noise process which is the result of independent events, such as photon arrival occurring at a constant rate, can be described by Poisson statistics [10]. The Poisson distribution was discussed in Section 2.2.7, and was presented in equation (2.5). Drawing samples from a Poisson distribution is a function implemented by Python's numerical programming library `NumPy`, which uses equation (2.5) rewritten as

$$f(k;\lambda) = \frac{\lambda^k e^{-\lambda}}{k!}, \tag{3.6}$$

which describes the probability of k events occurring within the observed interval λ, for events with an expected separation λ [79]. The function takes λ as argument, and for the dark current case λ is the calculated dark current electrons per pixel. The result is a matrix of the same size as the sensor, filled with dark current noise. This dark current matrix is added to the simulated sensor meaning the noise is added to the signal data.

Sky background

The sky background was introduced in Section 2.4.4 and consists of several different sources. Sky background is proportional to the exposure time [78], and in the simulation tool the user inputs the desired amount of *sky counts*, which is the expected number of generated electrons per second per pixel from the sky background. A higher sky count therefore signifies a noisier sky, and this parameter can be adjusted by the user to simulate a brighter Moon or more light pollution for example.

The sky background also follows a Poisson distribution [10], and the implementation of it is therefore very similar to to the implementation of the dark current. The same Poisson function from `NumPy` is used to draw samples from the Poisson distribution, and the corresponding λ is calculated as sky counts times exposure time, which equals to the total number of generated electrons per pixel from the sky background. The result is a matrix filled with noise from the sky background, which is added to the simulated sensor data.

Read noise

Read noise was introduced in Section 2.2.3 and arises during the electronic readout process of the sensor due to the electronic amplification of the signal. As for the dark current, read noise data is found in the sensor data sheet as a number of electrons and this parameter is inputted by the user. Read noise has a Gaussian distribution, and the standard deviation of the Gaussian is the read noise divided by the *gain* of the sensor. Gain controls the amplification of the electronic signal, but this parameter has not been implemented in the simulation tool. Instead, the read noise is directly used as the standard deviation of the Gaussian distribution which implies that gain = 1 is assumed.

Similarly as for the previous disturbances, the drawing of samples was implemented by using a `NumPy` function which describes the probability density for the Gaussian distribution as

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \tag{3.7}$$

where μ is the mean and σ the standard deviation [80]. The function takes σ as argument, which is inputted as equal to the read noise parameter. The result is a matrix filled with read noise, which is added to the the simulated sensor data.

A noteworthy point is that while the Poisson distribution only provides positive values, the Gaussian distribution provides both positive and negative values. This means the value of a pixel could be updated to equal a negative value, which does not make physical sense since these values represent a number of electrons. This has been corrected and is implemented once all disturbances have been added to the simulator.

Dark and hot pixels

Dark and hot pixels were introduced in Section 2.2.3, and as explained dark pixels are pixels which have a significantly lower response than their neighbours, less than 75% the response of the average pixel. Hot pixels instead have a much higher dark current than their neighbours, around 50 times higher than specified.

Dark pixels have been implemented by first allowing the user to specify the number of dark pixels to be used. This number is either known, or must be estimated. The dark pixel function then generates random numbers using a `NumPy` function [81], which corresponds to the $x-$ and y-coordinates of the dark pixels. In a real sensor, the locations of the dark pixels will not change, and the random numbers are therefore generated using a *seed* which makes sure the same indices are generated, but appears randomly distributed. The user is therefore required to provide a seed number for the sensor which can be any integer, but the same integer should be used if simulating the same sensor between different runs. Once the indices of the dark pixels have been generated, their values are decreased by 50% to simulate the low response.

The indices of the hot pixels are generated using the same strategy as for the dark pixels, but a different random generator seed is required which is inputted by the user. To know what to update the hot pixel values with, the dark current is calculated as earlier by multiplying the dark current number by the exposure time. This number is then multiplied by 50 to simulate the hot response of the pixel, in accordance with the theory presented in Section 2.2.3. This value, called hot current, is then added to the previous values of the hot pixels.

Checking the minimum and maximum sensor value

A final step is required to complete the modelling. Since the Gaussian function used to simulate the read noise may provide both positive and negative values, it is possible that some pixels have acquired negative values during the simulation which does not make physical sense. The simulation tool therefore checks all the values in the sensor, and if any negative value is found it updates the value to zero.

A similar check is performed for the maximum possible value of a pixel which is the well depth. The sensor again checks all the values of the sensor, and if a value larger than the well depth of the sensor is found it updates it to equal the well depth that was inputted by the user. If this step had been implemented when first adding the electron values to the sensor, the resulting sensor would contain too few electrons in total. This is due to the fact that the spreading by the atmosphere and telescope, which lowers the amount of electrons in the original pixel, is implemented afterwards in the simulator. In reality, the photons which give raise to electrons would have been spread already upon arrival to the sensor.

3.3.11 Modelling binning

In Section 2.2.4, the operational parameter binning was discussed which was further illustrated in Figure 2.8. When binning is used in the simulation, the user provides the binning parameter which denotes the number of pixels in one side of the super pixel. Providing a binning parameter that equals one therefore means that binning should not be used at all.

If binning is to be applied, the sensor is sent to the binning function in the simulation tool. The function starts by checking whether the number of pixels in one side of the sensor is divisible by the binning parameter. If the sensor side is not divisible, the function removes the corresponding number of rows or columns required to make the side divisible. The function then forms a 4D matrix (a matrix of matrices) where each sub-matrix corresponds to a super pixel. Each sub-matrix is then summarised to one pixel, and the function returns the binned sensor.

3.3.12 Defining the outputs of the sensor

The last step of the simulation process is to output and save the results. Several different types of outputs are generated, and these are described in this section. Full examples of the outputs can be seen in Section 4.1.2.

Outputs in the console

Some helpful information is outputted directly in the console and are printed as the simulation process runs. The first output is a message about which observation location is used, and the second output prints out which object is being observed. This is followed by important information about the satellite at the starting time of the observation: altitude, azimuth, distance, right ascension, and declination. If any of the warning messages discussed in Section 3.3.6 are triggered, they are printed next. The last print message informs the user about how many seconds it took to run the simulation.

Outputted figures

Two figures are outputted directly when the simulator has finished. The first outputted figure is the simulated sensor. This figure has tick marks along the axes to indicate the pixel indices, and a colour bar to see what electron values the different colours correspond to. The title contains information about how many pixels the sensor has (this is not equal to the user input if binning has been applied), the binning parameter, FOV, limiting magnitude chosen by the user, exposure time, and time step. Additionally, the satellite name, observation location, and starting time of the observation are also included in the title. This figure is not saved automatically, but the user can obviously choose to save it manually.

It should be noted that for most observational cases it is difficult to see anything in the sensor figure except the satellite on a black background. This is since the satellite is often the brightest object in the figure. The stars are visible if the user zooms in on them, and this process can be facilitated by first looking at the help image to know where to look. If the user wants to examine fine details of the image the black and white levels need to be adjusted. Depending on how the images and outputs from the simulator are to be used, this might want to be examined further in the future.

If the user chose to see the help figure, this is outputted as the second figure. The title of the figure contains information about which object is observed, at which time the observation takes place, which location the observation is from, and the altitude, azimuth, and distance to the satellite. Additionally, the title also includes the FOV, the magnitude limit that was chosen by the user, the exposure time, and the time step. As for the sensor image, this figure is not saved automatically but can be saved manually.

Saving the simulated sensor as a FITS file

The FITS format was discussed in Section 2.2.6 and is a way of storing the sensor data without compressing it. This has been implemented in the simulation tool by using functions from `AstroPy` [82]. If the users enables FITS storing, the sensor in a FITS file where the sensor data is stored in binary form, and the header contains all the information the user inputted and additionally some calculated values, as the FOV for example. The FITS file can be opened and examined using, for example, the free software program `ESA/ESO/NASA FITS Liberator` [83]. This software allows adjustment of the white and dark levels for example, which can be used to examine different details of the image.

Saving the simulated sensor as a PNG file

If the user enables saving the file in PNG format, the data as it was seen in the outputted figure is saved in a PNG file. This file only contains the sensor data image itself, without tick marks, title, or the colour bar. However, knowing which inputs that were used to generate the image is still valuable information to the user, and to accommodate this need all the information which was stored in the FITS header is outputted in an additional text file called *PNG log file*.

Chapter 4

Results and discussion

4.1 Reconstruction of a real image

The simulated images were verified in different scenarios by comparing them to real imagery provided by the Small Aperture Robotic Telescope Network (SMARTnet™) [24, 84]. Inputs from these images are provided to the simulation tool which performs forward modelling to generate the expected observation outcome.

SMARTnet™has been set up by the German Space Operation Center (GSOC), together with the Astronomical Institute of the University of Bern (AIUB). The main objective of the network is the free exchange of all gathered information within the involved partners. This information mainly consists of tracklet observations. SMARTnet™is currently operating two telescope stations, where one is located in Zimmerwald in Switzerland, and the second station in Sutherland in South Africa [84].

The real image which will be attempted to be replicated is seen in Figure 4.1. This image has been created by opening the original FITS file in `ESA/ESO/NASA FITS Liberator`, and saving it as a Tag Image File Format (TIFF) file which applies some automatic scaling. The observation was performed by using satellite tracking, and the satellite is seen as a small dot at the mid-left of the image, indicated by an orange arrow. Some star tracklets are clearly seen in the image as well. One known difference is that the position of the satellite will differ in the images. This is since the satellite in the real image is located far to the left, while the tool simulates the observed object as the center of the image. However, an overlap is expected between the images and the star tracklets in this overlap will be compared, as well as the coordinates in the equatorial system, in order to evaluate the resemblance of the images.

Figure 4.1: A real image taken by using satellite tracking. The orange arrow indicated the satellite, seen as a small dot. This image was provided by SMARTnet™.

4.1.1 Inputs

In order to replicate the image, several different inputs are required. Information about these inputs are gathered from several different sources. Some inputs can be taken directly from these sources, but some need to be estimated and the corresponding assumptions motivated.

Observation location data

The observation location of the real image was the telescope station in Sutherland, South Africa. This station is placed at the South African Astronomical Observatory (SAAO), where the latitude equals -32.38072° and the longitude equals 20.81078°. The elevation of the location is 1761 MSL. These inputs are provided to the simulation tool by a text file containing all the relevant information. The name of the file

```
OBSERVATION LOCATION DATA
If coordinates are given in compass direction, replace as: N=1, S=-1, E=1, W=-1

Long name, short name, latitude (north/south), longitude (east/west), elevation [meters above sea level]
South African Astronomical Observatory, SAAO, -32.38072, 20.81078, 1761
```

Figure 4.2: Input format of the observation location data. The name of the file is provided as input to the simulation tool.

is then given as input, and the tool uses this information to create the observation location name. The input file is seen in Figure 4.2.

Time

The real image was provided in the FITS format, and by examining the FITS header the input data regarding time is found. The header informs that the date of observation was the 22nd of February, and that the time was 18:04:40.780. It is not formally stated that the time is given in UTC, but this is assumed. South Africa does not apply daylight saving, and the local time zone is UTC+2 [85]. The sunset on the 22nd of February occurred at 19:30 local time (17:30 UTC), and since the observation is assumed to be during the night it is reasonable that the provided time is in UTC. Additionally, the header also states that the exposure time was 5.074937 seconds.

The simulation tool also requires the time step (Δt) to be inputted. A fine enough time step can be found by running some test simulations, but by studying the image in Figure 4.2 it is seen that the tracklets are not too long, and no too many stars are visible in the frame. The time step is therefore set to 0.005, which means around 1000 coordinate pairs will be calculated for each visible star. This was proven to be more than enough to ensure continuous star tracklets, while still keeping the run time within practical limits.

Sensor parameters

The FITS header states that the camera used in the observation is called FLI16803, which is assumed to be the FLI ProLine CCD camera KAF-16803. The data sheet for this camera [86] gives that the CCD sensor consists of 4096×4096 pixels, with a pixel size of 9 µm (assuming square pixels), and a full well capacity of 100,000 electrons.

The FITS header says that the temperature of the chip during the observation was -20.00° C, and the data sheet says that the corresponding dark current is < 0.005 electrons per second. The simulation tool requires the dark current input to be in electrons per pixel per seconds. The dark current is therefore re-calculated as $0.005/(4096 \times 4096) \approx 2.98 \times 10^{-10}$ electrons/second/pixel. The read noise is given by the data sheet to be 10 electrons for the CCD camera.

Examining the quantum efficiency curve in the sensor data sheet, it is found that the $Q_{\text{eff}} \approx 0.4$ at a wavelength equal to 700 nm, and the passband is very

roughly approximated to be 200 nm. The wavelength and passband used in the real observations are not known. The peak Q_{eff} of the sensor is 0.6, and the passband with $Q_{\text{eff}} > 0.5$ is around 200 nm.

Telescope parameters

The FITS header states that the telescope is the CDK20. According to the data sheet, this telescope has an aperture of 508 mm and a focal length of 3454 mm [87]. This means that the focal number is 6.8, but the simulator tool will calculate this itself. The data sheet also states that the central obscuration is 39% of the central mirror. The efficiency of the telescope at 700 nm is not known. However, the data sheet states that the coatings of both the primary and secondary mirror are 96%. The total efficiency is therefore assumed to be $0.96 \times 0.96 \approx 0.92$.

Similarly as for the observation location data, both the sensor and telescope parameters are saved in a file. The file name is then provided as input to the simulation tool. An example of the input file is seen in Figure 4.3. Also here, it is important to write the data in the specified order and on the specified line, and also to separate the data by a comma and a space.

```
SENSOR DATA
Sensor width [n.o. pixels], Sensor height [n.o. pixels], Pixel width [m], Pixel height [m]
4096, 4096, 9e-6, 9e-6

Well depth [e-], Quantum efficiency, Read noise [e-], Dark current [e-/pix/sec](@-20°C), Wavelength passband [m]
100000, 0.4, 10, 2.98e-10, 200e-9

Dark pixel seed (any integer), hot pixel seed (any other integer)
19920817, 19950820

TELESCOPE DATA
Aperture diameter [m], Focal length [m], Obscuration [fraction of diameter], Optics efficiency
508e-3, 3454e-3, 0.39, 0.9
```

Figure 4.3: Input format for the sensor and telescope parameters. The name of the file is provided as input to the simulation tool.

At this stage, the seeds for the dark and hot pixels are also defined and inputted by loading them from the file. The values of the seeds do not matter, they can be any integers as long as they are not identical.

Satellite data and orbital information

Examining the FITS header further, it is found that the observed object name is 10022A. This seems to be a shortened version of the International Designator which was introduced in Table 2.5. Number 10 indicates that the satellite was launched in 2010, number 22 that it was the 22nd launch in 2010 and the letter that the sequential id is A. By searching on the number it is found that this satellite is the NAVSTAR 65, which is also called USA 213 or GPS 25, and has the NORAD ID 36585 [88]. It was launched on the 28th of May 2010, and its maximum distance to

Earth is 20,446.0 km. The estimated RCS is 6.4686 square meters, which is used as an input parameter directly in the simulation tool.

Since the simulated observational date is the 22nd of February, a TLE with an epoch close to this date is required. Historical TLEs can be requested from `http://celestrak.com/norad/archives/request.php`, and by doing so, all TLEs from February 2021 are provided. February 22nd is the 53rd day of the year, and the epoch therefore corresponds to 21053 in accordance with the definition which was also given in Table 2.5. Out of all the TLEs from February, the closest to this epoch is found to be 21052.23067455, meaning around 06:00 0n the 21st of February 2021. This is a likely epoch to use to simulate upcoming observations, and therefore this TLE is chosen as input to the simulation tool. The complete input file of the TLE and satellite name is seen in Figure 4.4. As for the previous cases, the file name is then provided as input to the simulation tool which loads the TLE data.

```
NAVSTAR 65 (USA 213 / GPS 25/ NORAD 36585 / 10022A)
1 36585U 10022A   21052.23067455  .00000018  00000-0  00000-0 0  9993
2 36585  55.1280 286.9166 0096427  53.3396 118.5570  2.00564651 78646
```

Figure 4.4: The inputted TLE data for the satellite NAVSTAR 65.

Having these data, the only remaining satellite parameter to input is the albedo of the satellite. In Section 2.3.3, it was stated that albedo values from 0.175 to 0.2 have been used when modelling GEO satellites. The albedo is therefore, quite arbitrarily, chosen to equal 0.2.

Disturbances

The next step is to choose the disturbances to include in the simulation. Dark current and read noise are inevitable when doing a real observation, and are therefore included in the simulation. Some sky background counts is also to be expected, but the number of counts can be hard to predict. SAAO is a professional observation location, and little sky glow should therefore be expected. According to `www.lightpollutionmap.info`, the sky quality at SAAO is around 22 magnitudes per square arcsecond. By then estimating the sky background electron rate using `www.tools.sharpcap.co.uk`, it is estimated that the sky background should give raise to around 1.58 electrons/pixel/second. However, on the 22nd of February 2021 the Moon was up, and the full Moon occurred no later than five days after the observation date [85]. According to the FITS header, the observation started at coordinates RA = 120.041343°, and DEC = -3.092003°. Computing the corresponding coordinates of the Moon at this time by using Skyfield, it is found that RA = 97.986°, and DEC = 25.808° which is not too far away from the observed location. The sky background count should therefore be increased by quite a lot, and is therefore set to 20 electrons/pixel/second.

The atmospheric efficiency also needs to be estimated. We assume that this was a cloud free night, but the light transmission through the atmosphere also depends

on the observed wavelength. For an observation of 700 nm, the transmittance is about 0.8 [89], which therefore is the number used for the atmospheric efficiency input.

Examining the real image further in `ESA/ESO/NASA FITS Liberator`, it is seen that it contains hot pixels which are seen as brighter spots in Figure 4.5 and are marked by orange squares. The image is a zoomed in and cropped version of Figure 4.1, the satellite is seen in the upper left corner. It is also seen that the hot pixels vary in brightness, where the two pixels to the right are brighter than the other two pixels.

Figure 4.5: Example of hot pixels in the real SMARTnet™image. The hot pixels are marked by orange squares, and can be seen to vary in brightness.

The simulation tool requires the user to input how many hot pixels to add to the simulated sensor. A very rough estimation from the FITS image indicates that it contains around 200 hot pixels in total. The dark pixels are not as distinguishable as the hot pixels, but since dark and hot pixels both form due to contamination of the sensor it seems reasonable to assume around the same amount of dark pixels is present in the sensor as well. Both the input parameters for the number of dark and hot pixels are therefore set to 200.

Spread of signal

Both PSF and seeing will affect the spread of the light, and both the Airy disk and atmosphere parameters are therefore set to True. The tool will calculate the input radius of the Airy disk itself, but the *sigma* parameter is required to be inputted

by the user. This parameter corresponds to how much the light will spread due to the seeing at the location and the specific atmospheric conditions during the observation time. SAAO is located at 1761 MSL which will improve the seeing, and this means sigma should be small. For this simulation the value of 0.5 is chosen.

Image formats

The image format inputs which formats to save the final simulated sensor in. For this simulation both the FITS file and PNG formats are requested, and both these parameters are therefore set to True. This will result in that the simulator automatically saves the sensor to a FITS file, a PNG file, and additionally creates a text file containing all the user inputs.

Additional inputs

Some additional inputs are left to be chosen. The satellite tracking parameter should obviously be set to True since the real image was created by satellite tracking. The limiting magnitude parameter is set to 16 which currently is the maximum possible value in the simulation tool. Stars as faint as magnitude 16 will probably not be visible in the resulting image due to the short exposure time and the use of satellite tracking, but nevertheless this ensures full possible coverage.

By re-examining the FITS header of the real image, it is found that both the binning in x and y were set to one during the real observation. Therefore, the simulation tool binning parameter is also set to one. The last input is then to decide whether to output the help image. For this example the help image is desired, and this parameter is therefore set to True.

Summary of inputs

All inputs are now defined, and a summary of all the different parameters is presented in Table 4.1.

Table 4.1: Summary of all the inputs used to replicate the real image.

Name	**Value**	**Unit**
Observation start time	2021, 2, 22, 18, 4, 40.780	Year, month, day, hour, minute, second [UTC]
Exposure time	5.074937	Seconds
Time step	0.005	Seconds
Location name	South African Astronomical Observatory, SAAO	-
Latitude and longitude	-32.38072, 20.81078	Degrees
CelesTrak TLE	False	Boolean
TLE	*See Figure 4.4*	-
Satellite RCS	6.4686	Square meters
Satellite albedo	0.2	-
Sensor width and height	4096×4096	Number of pixels
Pixel width (and height)	9×10^{-6}	Meters
Well depth	100,000	Electrons
Quantum efficiency	0.85	-
Read noise	10	Electrons
Dark current (@-20°C)	2.98×10^{-10}	Electrons/pixel/second
Passband for wavelength	200×10^{-9}	Meters
Dark and hot pixel seeds	19920817, 19950820	-
Aperture diameter	0.508	Meters
Focal length	3.454	Meters
Obscuration	0.39	Fraction of diameter
Optics efficiency	0.92	-
Limiting magnitude	16	-
Satellite tracking	True	Boolean
Binning parameter	1	Number of pixels
Atmospheric efficiency	0.8	-
Apply dark current	True	Boolean
Apply read noise	True	Boolean
Sky background counts	20	Electrons/pixel/second
Dark pixels	200	Number of pixels
Hot pixels	200	Number of pixels
Apply Airy disk	True	Boolean
Apply atmosphere	True	Boolean
Sigma	0.5	-
Save as FITS file	True	Boolean
Save as PNG file	True	Boolean
Help image	True	Boolean

4.1.2 Outputs

The defined inputs are inputted to the simulation tool, which runs the simulation. This section presents the results and different outputs, as well as some parameters that are calculated by the tool.

Outputs in the console

The first output consists of messages in the console, and they are presented in Figure 4.6. As seen, the first message tells the user which observation location the simulation tool simulated the observation from. The second message tells the user which satellite is being observed. An attentive observer can see that that the "/" characters that were used in the input file shown in Figure 4.4 have been replaced by the character "-". This is done by the simulation tool since the satellite name is used to create the names of the saved files, and using the character "/" in the file name is not allowed.

```
You are observing from South African Astronomical Observatory (SAAO).
You are observing the satellite NAVSTAR 65 (USA 213 - GPS 25- NORAD 36585 - 10022A).
At 2021/02/22 18.04.41 ; altitude = 45.42 °,     azimuth = 56.40 °, distance =
21831.60 km. Right ascension =  8.03 hours, declination =  -3.00 ° .
It took 697.72 seconds to run.
```

Figure 4.6: Resulting outputs in the console window from the replication simulation run.

Next follows information about the coordinates of the satellite at the start of the observation. These are given in both the horizontal and equatorial coordinate system. The distance to the satellite is also provided.

The last message in the console consists of the time it took to run the simulation. In this particular case it was 697.72 seconds, which equals to about 11 minutes and 37 seconds. This is quite long, mostly due to the high limiting magnitude and the fine time step which is used in the simulation. By examining the reduced star catalogues it is seen that in total 1097 stars were considered to be able to enter the frame. Three of these stars are from the Hipparcos catalogue, and 1094 are from the GAIA EDR3 catalogue. A closer inspection of the reduced catalogues coordinates implies that most likely, the three stars from Hipparcos are also included in the GAIA EDR3, which will have resulted in a slight error which was discussed in Section 3.3.3. In total, 1016 stereographic coordinate pairs were calculated for each star.

It should also be noted that, as expected, none of the three warning messages were printed for this simulation. This means that during the observation the satellite was sunlit, the sun was down at the observation location, and as seen from the altitude coordinate the satellite was above the horizon.

Outputted figures

The next type of outputs are the outputted figures from the simulation tool. The first one is the simulated sensor which is seen in Figure 4.7. The title contains some of the input information such as the sensor size, limiting magnitude, exposure time, and Δt (plot interval), but also the FOV which has been calculated to $0.61^\circ \times 0.61^\circ$. The third line in the title informs the user which object is shown, from which location, and at which time the observation was initiated.

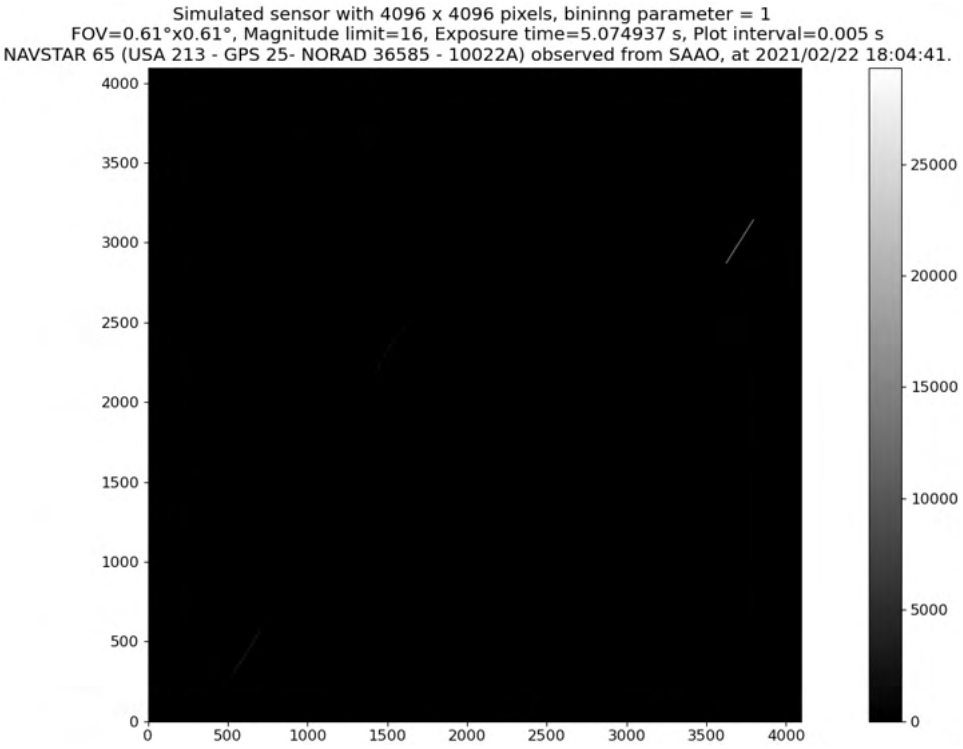


Figure 4.7: Resulting unprocessed figure of the simulated sensor. Due to the current scaling applied by the software it is not possible to see much except from a bright star tracklet to the right in the image.

As mentioned in Section 3.3.12, it is difficult too see much in this image besides the very brightest parts. In this case, it is a star tracklet which can be seen to the right in the image. By again comparing with the reduced star catalogues, it is seen that the brightest star in the image is of magnitude 6.458889 according to GAIA EDR3. This is one of the stars which overlaps with the Hipparcos catalogue, which has estimated the star as being of magnitude 6.41. The star should therefore be about double as bright in the resulting image due to the double plotting,

meaning it in the image corresponds to a star of around magnitude 5.68, based on equation (2.18).

The last piece of information from the sensor image in Figure 4.7 comes from the colour bar seen to the right. The well depth of the sensor was 100,000 electrons, but the bar only goes up to about 30,000. This means that for this particular observation with the specified seeing and system parameters, no pixels were saturated. When the same simulation was ran without applying any type of spread, it was seen that some pixels did saturate. The corresponding resulting image is seen in Section B.1, Figure B.1.

The second outputted image from the simulation tool is the help image, which is seen in Figure 4.8. As for the sensor image, the title also contains some of the inputted parameters and calculated parameters, including the initial coordinates of the satellite in the horizontal coordinate system. The green line in the upper left corner is a constellation line. The help image shows clearly all the star tracklets which are located in the frame, while the satellite in the center can hardly be discerned.

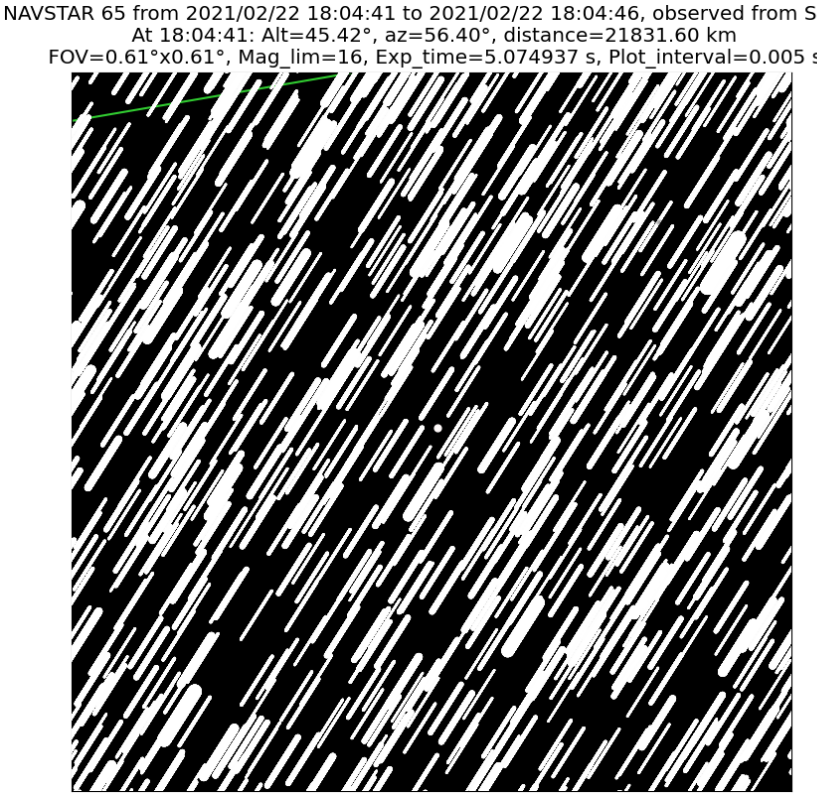


Figure 4.8: Resulting outputted help image from the reconstruction simulation.

The saved PNG file and associated log file

Since one of the user input was to save the result as a PNG file, the simulation tool does so. The PNG file itself is just a stripped version of the sensor image that was seen in Figure 4.7, and the resulting PNG image is seen in Section B.2, Figure B.2. An associated text log file is created and saved in addition to the PNG file. This log file contains most of the user inputs, and the resulting file from the reconstruction run is also seen in Section B.2, Figure B.3.

The FITS file

Since the input to save the file as a FITS file was given, the simulator tool does so too. The FITS file header contains all the information that was given in the log text file, and a snippet of it is seen in Section B.3, Figure B.4. The result from opening the FITS file in `ESA/ESO/NASA FITS Liberator`, and saving it as a TIFF file which applies some automatic scaling, can be seen in Figure 4.9. The satellite is seen as a small dot in the middle of the image.

By adjusting the black and white levels of the FITS file in `ESA/ESO/NASA FITS Liberator`, different details of the image can be enhanced. In Figure 4.10, the white level has been decreased and the FITS file has been zoomed in on the satellite to show the spread of the satellite electrons. In this image, the white level has been lowered enough so that the noise is visible as well.

When looking at Figure 4.9, it appears as some of the star tracklets might consist of discrete points instead of being continuous tracklet. This is not true, and arises from the computer not being able to match the resolution of the image. When zooming in on the tracklets, even the faintest of them are shown to be continuous. This is further illustrated in Figure 4.11, where the white levels has been even further decreased to show very faint tracklets. It can also be seen how some of the faintest tracklets are almost disappearing into the background noise.

Figure 4.9: The simulated SMARTnet™image. This image has been processed by adjusting the black and white levels.

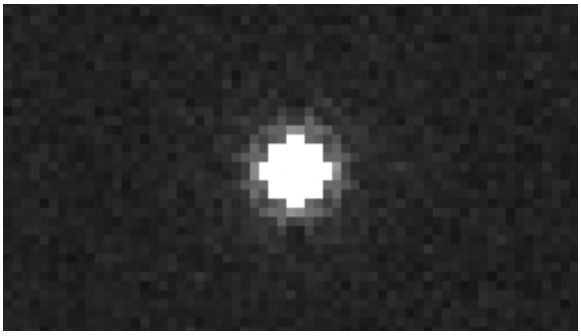


Figure 4.10: The resulting satellite in the simulated image. The white level of the FITS file has been decreased to enhance the spread to the nearby pixels due to PSF and seeing.

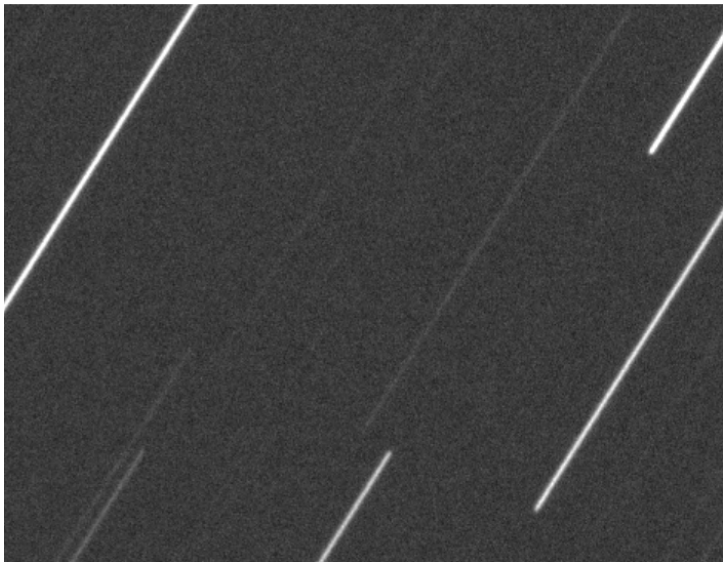


Figure 4.11: When zooming in on the satellite tracklets in the simulated image they are shown to be continuous. Here, the white level of the image has been lowered to be able to see even the faintest star tracklets.

Summary of some calculated parameters

Some of the calculated parameters, such as the coordinates for example, have already been presented. These parameters, together with some additional parameters calculated by the simulation tool, are summarised in Table 4.2.

Table 4.2: Summary of some parameters calculated by the simulation tool.

Name	**Value**	**Unit**
Airy disk radius	0.64517	Number of pixels
Altitude	45.42	Degrees
Azimuth	56.40	Degrees
RA	120.413712	Degrees
DEC	-3.001932	Degrees
Diffraction limit	1.6811×10^{-6}	Radians
Distance	21831.60	Kilometers
Effective telescope area	0.1719	Square meters
Focal number	6.799	-
FOV	0.612×0.612	Degrees
Satellite apparent magnitude	11.574	-

4.1.3 Comparison with the real image

Having acquired the simulated outputs, it is now time to do some comparisons between the simulated and the real images. The comparisons consist of both visual inspections, and a comparison of the equatorial coordinates.

Side by side comparisons

In all side by side comparisons, the simulated image is located to the left, and the real image is to the right. The first side by side comparison consists of placing the complete images next to each other, and this comparison is seen in Figure 4.12.

Figure 4.12: Comparison of the simulated and real images. The left image is the simulated image, and the right image is the real image.

The most noticeable difference is that the background of the real image is brighter than the background of the simulated image. This depends partly on how the greys of the images have been scaled, but it is more likely that the real image had a higher sky background count than what was chosen for the simulated image. This hypotheses is strengthened by the fact that the real image was taken at a small angular distance from the Moon, on a night in the lunar cycle when the Moon was five days from being full. The real image also has a gradient due to the disuniformity in illumination. This type of background behaviour can either be caused by the image being uncalibrated, or by internal reflection caused by the Moon.

The next side by side comparison is shown in Figure 4.13 and also consists of the same two images next to each other, but the images are now marked in order to better illustrate the positions of the satellite and different star tracklets. The satellite in each image has been marked by a red square, and corresponding

star tracklets are marked by coloured circles. As seen, the tracklets correspond well within the overlap of the images. Brighter tracklets in the real image are also brighter in the simulated image, and from visual inspection no tracklet is missing. The simulated image contains more of the faint tracklets. This is due to the background of the real image being much brighter, which means some of the faintest tracklets disappear into the background noise.

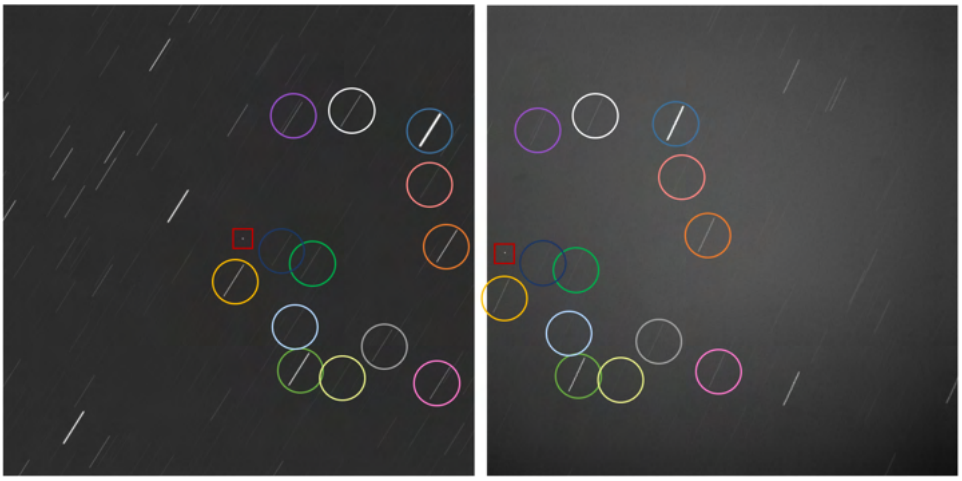


Figure 4.13: Marked comparison of the simulated and real images. The left image is the simulated image, and the right image is the real image. The satellite in each image is marked by a red square, and corresponding star tracklets are marked by coloured circles.

It is also noticeable that the positions of the satellite in each image match very well. As a reminder, the satellite of the position in the simulated image has been calculated from propagating a TLE. Some difference in the positions could therefore have been explained by how the TLEs are propagated.

By close inspection of the comparisons, it can also been seen that there is some difference in the inclination of the star tracklets. This is assumed to be due to the stereographic projection function from `Skyfield` which determines the rotation of the FOV, which also does not take the type of telescope mounting into account. This difference in inclination might therefore occur due to the difference in how the projection is built versus the mounting of the telescope, which for this case is unknown.

Figure 4.14 shows a side by side comparison zoomed in on the satellite from the real and simulated images. The FITS files black and white levels have been scaled to display the same levels of background brightness, in order to show the spread of the satellite electrons among the nearby pixels.

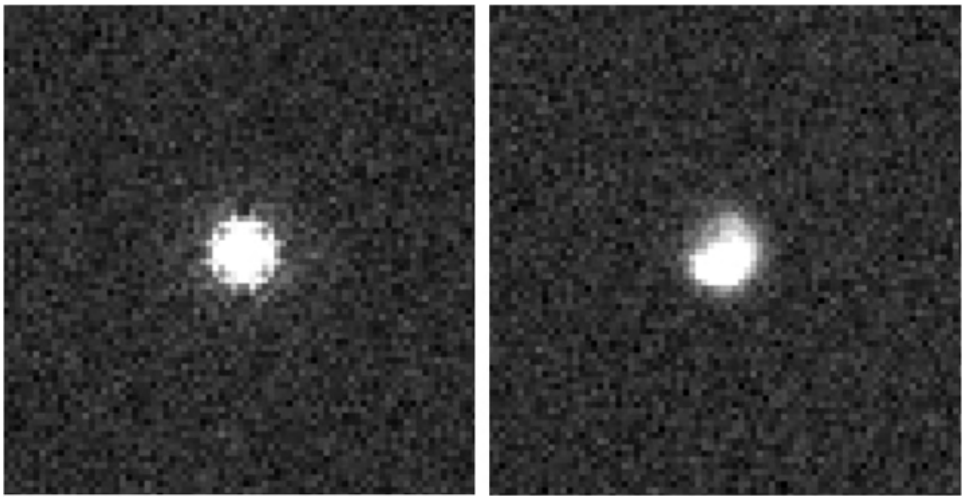


Figure 4.14: Side by side comparison of the satellites. The left image is the simulated image, and the right image is the real image. The backgrounds have been scaled to display equal levels of brightness.

As seen from Figure 4.14, the satellites are about the same size which confirms the apparent magnitude calculation and the PSF assumptions. In the simulated image, the satellite is symmetric which will always be the case due to how the simulation tool has been designed. The real satellite is not symmetric, and this is most likely due to a lens disturbance. This hypotheses is strengthened by the fact that the satellite in the real image is located far towards the edge of the FOV where lens disturbances are most noticeable. It can also be seen that the pattern of the background noise looks very similar between the images.

In Figure 4.15, a side by side comparison of some tracklets is shown. The bright star tracklet located in the bottom middle of the image is the tracklet which was marked by a light green circle in Figure 4.13, and the exact position of this cutout from the full image is seen in Section B.3, Figure B.5.

The images in Figure 4.15 have been scaled to display similar levels of brightness, and it can again be seen that the real image has a background brightness gradient. Also, as was discussed previously, the real image displays more faint star tracklets since the background is darker compared to the real image, where faint tracklets disappear into the background noise. The change in inclination is also further enhanced when comparing the images in this zoomed in version. The tracklets themselves display similar behaviours in terms of how the brighter tracklets appear clearer than the fainter tracklets, which are more grey in colour. A noticeable point is how the brightest tracklet appears jagged in the real image, compared to the smoother behaviour of the corresponding simulated tracklet. This could be an effect of mount disturbances, caused by wind for example. The same simulated star

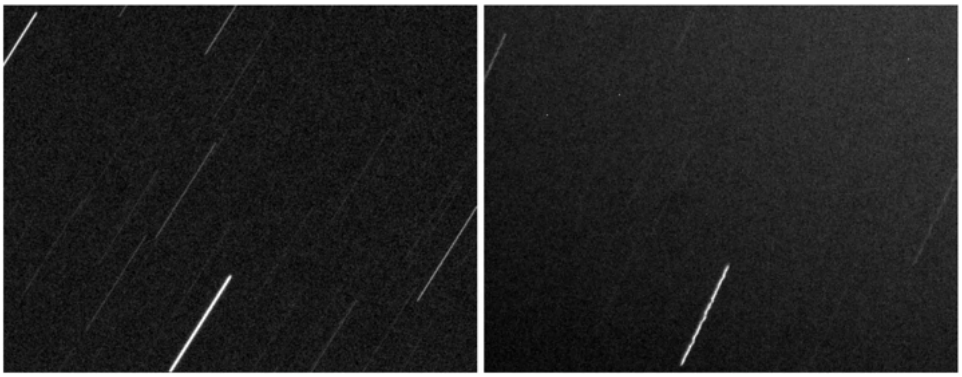


Figure 4.15: Side by side comparison of some star tracklets. The left image is the simulated image, and the right image is the real image. The backgrounds have been scaled to display similar levels of brightness.

tracklet also appears more spread out compared to the real one. This can partly depend on the scaling of the white and black levels, but also on the estimated sigma parameter and limitations in the PSF modelling.

In the real image, it is possible to see some hot pixels. When examining the simulated image, no obvious hot pixels were found. This could mean that the dark current response of the real sensor is higher than the simulated response, and this parameter could be adjusted to more clearly simulate hot pixels.

Comparison of equatorial coordinates

The calculated equatorial coordinates for the satellite position are seen in Table 4.3, together with the real observation coordinates found in the FITS header. As seen, they are quite similar and the difference is about 0.37 degrees for the RA, and 0.09 degrees for the DEC. However, the coordinates from the real image most likely correspond to the coordinates of the center of the image, and not the satellite itself. This means the real image coordinates should correspond to a location close to the tracklet marked by an orange circle in Figure 4.13 in the simulated image. This offset is a possible explanation for the difference of the coordinates.

Table 4.3: Comparison of the equatorial coordinates.

Name	**Unit**	**Value, simulation**	**Value, observation**	**Δ**
RA	Degrees	120.413712	120.041343	0.372369
DEC	Degrees	-3.001932	-3.092003	0.090071

4.2 A second image reconstruction

In order to further evaluate the performance of the simulation tool, a second real image is attempted to be reconstructed. This image can be seen in Figure 4.16. As for the first reconstruction simulation, this image is also provided by SMARTnet™, and has been taken from the same observation location in South Africa. The observed object this time is the geosynchronous communications satellite INTELSAT 901.

Figure 4.16: Original SMARTnet™image used in the second reconstruction simulation. The satellite has been marked by a red square.

When examining the image closely, more than one satellite can be seen, which is not too surprising considering it is the geostationary belt that has been observed. The tracked satellite (most centered in the image), has been marked by a red square in Figure 4.16.

4.2.1 Inputs

The methodology to find and estimate the input parameters for this reconstruction follows the strategy in Section 4.1. Most of the parameters are found in the FITS file header. The resulting input parameters are seen in Table 4.4.

The observed satellite is INTELSAT 901, and the RCS of this satellite has been estimated to 15.9919 square meters [90]. The albedo of the satellite is left at 0.2 in accordance with the discussion about albedo in Section 4.1.1.

The observation was performed from the SAAO, on the 25th of March 2020, at 01:07:49.951. Year 2020 was a leap year, and this date was therefore the 85th day of the year. When examining the historical TLEs, it is found that the closest epoch before this date is 20082.2951473, meaning three days before the simulation. The complete TLE can be seen in Figure 4.17.

```
INTELSAT 901
1 26824U 01024A   20082.29517473 -.00000221  00000-0  00000+0 0  9994
2 26824   0.2217  90.5467 0003300 342.1628 203.0541  0.98664001 68743
```

Figure 4.17: The TLE used to reconstruct the second image.

The sensor is the same sensor (FLI ProLine CCD camera KAF-16803) used in the first simulation, and all the sensor parameters are therefore unchanged. However, the telescope is different. In the FITS header the name of the telescope is 'ASA_8H', and this is assumed to be the Astro Systeme Austria (ASA) Astrograph 8" H f 2.8 telescope. In the datasheet it is found that the aperture of the telescope is 200 mm, and the focal length is 0.560 mm, but it provides no information about the obstruction [91]. However, it is found that the diameter of the secondary mirror is 100 mm, and the obstruction is therefore assumed to be 100/200=0.5. No information about the efficiency is found, and it is therefore assumed to be 0.9.

When examining the lunar cycle, it is found that the 25th March 2020 was the day after the new moon [92]. It is also found that at 03:07 local time, the Moon was under the horizon. The sky background counts parameter is therefore set to 1.58 in accordance with the discussion in Section 4.1.1.

In the first reconstruction, the time step was set to 0.005 seconds. However, when inspecting the original image in Figure 4.16, it can be seen that FOV is much larger and the tracklets are much shorter. A larger FOV implies many more stars will be considered, and this fact, in combination with the shorter tracklets, motivates the decision to lower the time step. The time step is therefore set to 0.01, which will result in about 507 coordinate pairs for each object in the FOV.

Table 4.4: Summary of all the inputs used to replicate the second real image.

Name	**Value**	**Unit**
Observation start time	2020, 3, 25, 1, 7, 49.951	Year, month, day, hour, minute, second [UTC]
Exposure time	5.069727	Seconds
Time step	0.01	Seconds
Location name	South African Astronomical Observatory, SAAO	-
Latitude and longitude	-32.38072, 20.81078	Degrees
CelesTrak TLE	False	Boolean
TLE	*See Figure 4.17*	-
Satellite RCS	15.9919	Square meters
Satellite albedo	0.2	-
Sensor width and height	4096×4096	Number of pixels
Pixel width (and height)	9×10^{-6}	Meters
Well depth	100,000	Electrons
Quantum efficiency	0.85	-
Read noise	10	Electrons
Dark current (@-20°C)	2.98×10^{-10}	Electrons/pixel/second
Passband for wavelength	200×10^{-9}	Meters
Dark and hot pixel seeds	19920817, 19950820	-
Aperture diameter	0.200	Meters
Focal length	0.560	Meters
Obscuration	0.5	Fraction of diameter
Optics efficiency	0.9	-
Limiting magnitude	16	-
Satellite tracking	True	Boolean
Binning parameter	1	Number of pixels
Atmospheric efficiency	0.8	-
Apply dark current	True	Boolean
Apply read noise	True	Boolean
Sky background counts	1.58	Electrons/pixel/second
Dark pixels	200	Number of pixels
Hot pixels	200	Number of pixels
Apply Airy disk	True	Boolean
Apply atmosphere	True	Boolean
Sigma	0.5	-
Save as FITS file	True	Boolean
Save as PNG file	True	Boolean
Help image	True	Boolean

4.2.2 Outputs

The simulation tool is then run with the inputs from Table 4.4. The resulting full scale FITS image is seen in Figure 4.18. The satellite, which is barely visible, has been marked by a red square.

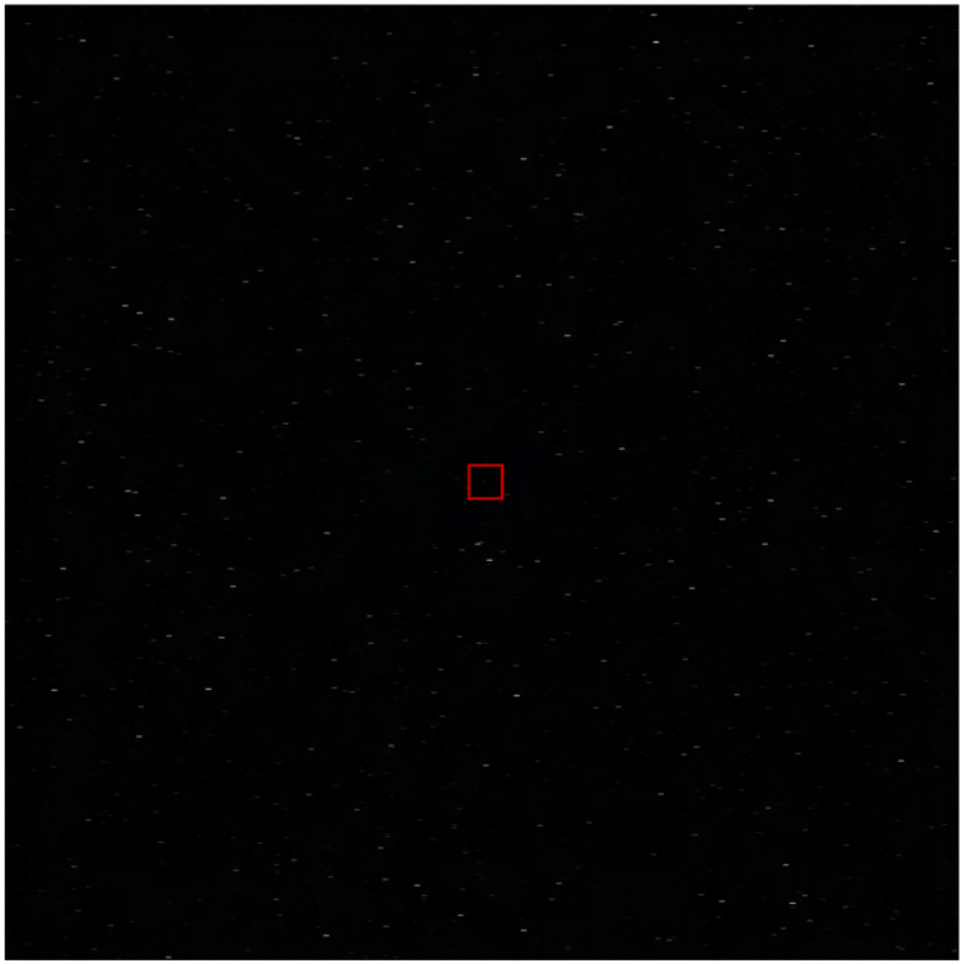


Figure 4.18: The resulting simulated image from the second reconstruction.

The corresponding output from the console is seen in Figure 4.19. As was seen for the first reconstruction simulation and expected also for this simulation, none of the warning messages have been triggered. The first outputted message is, as before, the observation location name which is followed by the name of the observed object. Then comes the observation date, and the coordinates of the satellite at the

start of the observation. The last message is the time it took to run the simulation. This particular case took 1154.11 seconds to run, which equals to about 19 minutes and 14 seconds. This is a very long run time, and is mostly due to the high limiting magnitude and the increased FOV, and therefore the increased number of stars to consider. In total, coordinates were calculated for 4592 stars, out of which 35 stars are from the Hipparcos catalogue, and 4557 stars are from the GAIA EDR3 catalogue.

```
You are observing from South African Astronomical Observatory (SAAO).
You are observing the satellite INTELSAT 901.
At 2020/03/25 01.07.50 ; altitude = 27.04 °,     azimuth = 296.36 °, distance = 39340.32 km.
Right ascension =  11.15 hours, declination =  5.30 ° .
It took 1154.11 seconds to run.
```

Figure 4.19: Resulting outputs in the console window from the second replication simulation run.

The sensor image that is directly outputted from the simulation tool is seen in Section C.1, Figure C.1. The corresponding help image, and the directly saved PNG file, are also seen in Section C.1 in Figure C.2 and Figure C.3, respectively. As for the first simulation, it is not possible to see much in the sensor and PNG images except a few of the very brightest tracklets. Even in the help image, it is hard to distinguish the satellite among all the short star tracklets. A summary of parameters which were calculated by the simulation tool is seen in Table 4.5.

Table 4.5: Summary of some parameters calculated by the simulation tool for the second simulation.

Name	**Value**	**Unit**
Airy disk radius	0.26569	Number of pixels
Altitude	27.04	Degrees
Azimuth	296.36	Degrees
RA	167.203016	Degrees
DEC	5.301359	Degrees
Diffraction limit	4.270×10^{-6}	Radians
Distance	39340.32	Kilometers
Effective telescope area	0.0236	Square meters
Focal number	2.80	-
FOV	3.772×3.772	Degrees
Satellite apparent magnitude	11.708	-

4.2.3 Comparison with the real image

Having acquired the simulated image, it is time to do some comparisons between the simulated and the real images. As for the first reconstruction, the comparisons consist of both visual inspections, and a comparison of the equatorial coordinates.

Side by side comparisons

Figure 4.20: Comparison of the simulated and real images. The left image is the simulated image, and the right image is the real image

In Figure 4.20, the images have been placed next to each other. Due to the large FOV and short tracklets it is difficult to see fine details. It can be seen that the images have more similar background noise levels compared to the first reconstruction example. Again, the images have been marked in order to facilitate the understanding of how the images are orientated. Overlapping star tracklets have been marked by coloured circles, and satellites by coloured squares. The marked image is seen in Figure 4.21, where they have been places on top of each other to display as many details as possibly. As mentioned previously, multiple satellites were found in the original SMARTnet™image. The assumed tracked satellite has been marked by a red square, and a secondary satellite has been marked by an orange square.

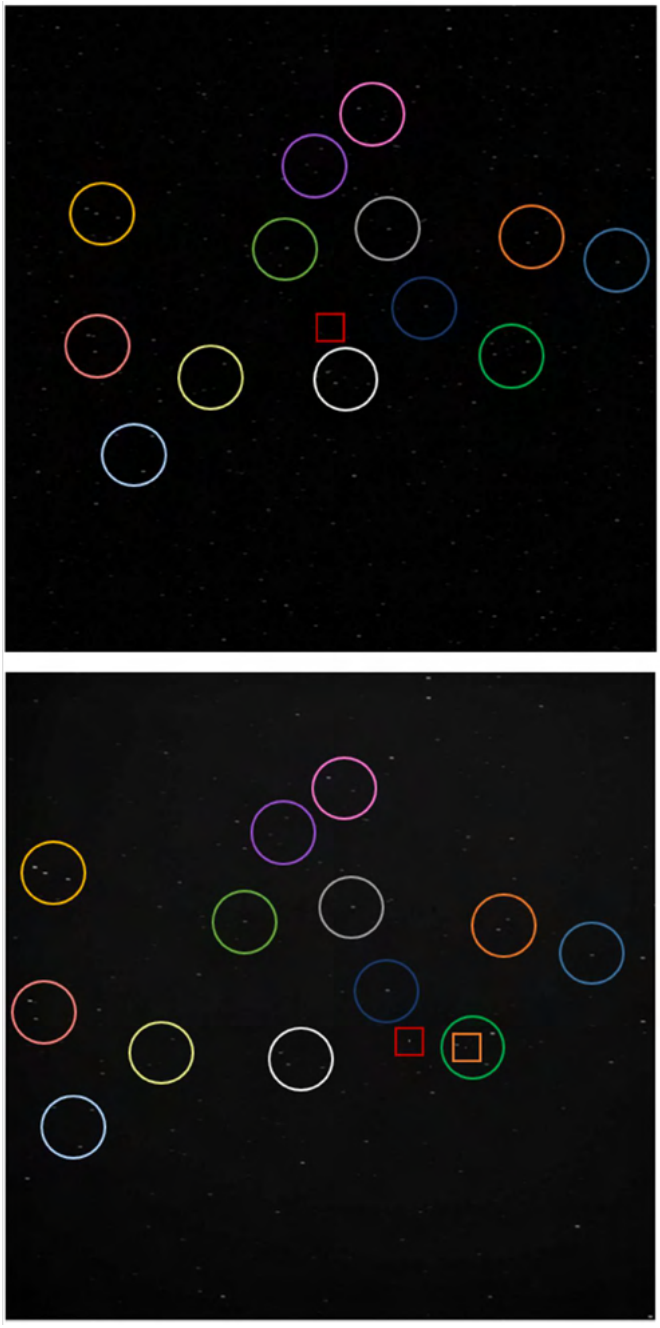


Figure 4.21: Marked comparison of the simulated and real images. The top image is the simulated image, and the bottom image is the real image. Corresponding star tracklets are marked by coloured circles, and the satellites are marked by coloured squares.

As seen from the marked comparison, the FOVs between the images correspond quite well. However, there is a clear discrepancy between the positions of the satellites that are supposed to be the same. Some discrepancy can be expected since the simulated satellite position has been calculated by propagating a three day old TLE, but for a geostationary satellite this is a suspiciously large discrepancy. This leads to the idea that there might be more satellites in the real image than the two that have been identified. When examining the original FITS file closer, no such extra satellite was found at the expected position, but the satellites are particularly difficult to spot in the real image due to the many star tracklets. The star tracklets are so short that quite some zooming in is required to be able to distinguish the satellites from the star tracklets. An additional satellite could therefore quite easily have been missed. In order to fully understand and conclude about this discrepancy in positions, further investigation would be required.

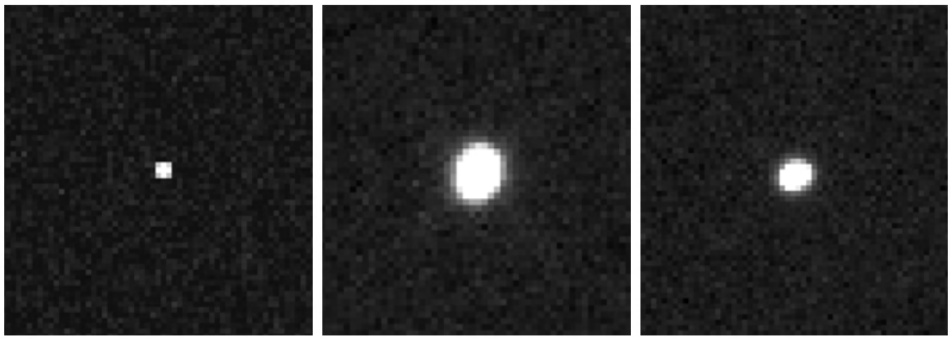


Figure 4.22: Comparison of the satellites. The left image is the simulated satellite, and the middle and the right images are real satellites.

In Figure 4.22, a side by side comparison of the satellites is seen. The left image is the simulated satellite, and the middle and right images are of the real satellites. The middle image corresponds to the satellite marked by a red square in the real image in Figure 4.13, and the right image corresponds to the satellite marked by an orange square. The images have been zoomed in at the same level, and the black and white levels have been slightly adjusted to show the same levels of brightness. This enhances that the behaviour of the background noise is very similar between the simulated and real image. In comparison to the first reconstruction, this real image does not have an equally strong sky background and no obvious gradient. This is most likely due to the difference between the lunar condition on the nights these images were captured.

It is noticeable how little the simulated satellite has been spread out compared to both of the real satellites. This could depend on many reasons. For example, the apparent magnitude of the satellite was calculated to 11.708 which is quite faint which might be due to the assumptions in the apparent magnitude model and the

estimated albedo. It could also depend on how the PSF has been modelled, as was seen in Table 4.5 the Airy disk radius was around 0.266 pixels which is very small. It could be that a better PSF model would have spread out the pixels more. It might also depend on the choice of sigma, since a higher value of sigma also would have spread out the satellite electrons to more nearby pixels.

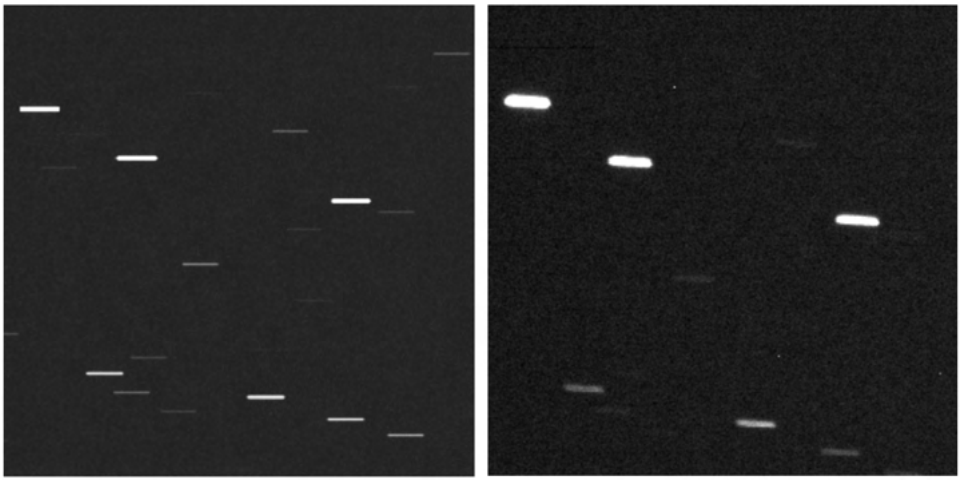


Figure 4.23: Side by side comparison of some star tracklets. The left image is the simulated image, and the right image is the real image. The backgrounds have been slightly scaled to display similar levels of brightness.

The last side by side comparison is seen in Figure 4.23, and shows the comparison of some star tracklets. The position of these images in the corresponding full image has been marked by a pink square in Figure C.4, which is found in Section C.2. Just as in the satellite comparison, it is noticeable that the simulated tracklets are much less spread out compared to the real tracklets. Also, more of the faint tracklets are visible in the simulated image. This could also be an effect of the low spread, since a higher spread would mean less visible tracklets. As was seen in the first reconstruction simulation, there is a difference in the inclination of the tracklets here as well. Some hot pixels can also be seen in the real image, and no equally distinguished hot pixels have been found in the simulated image. This reinforces further the theory that the dark current response parameter requires adjustment.

Comparison of equatorial coordinates

The calculated equatorial coordinates for the satellite position are seen in Table 4.6, together with the real observation coordinates which are found in the FITS header. The RA coordinates are very similar and the difference is about 0.18 degrees. The difference in DEC is about 1.038 degrees, which is bigger than expected since the

FOV is 3.772×3.772 degrees, meaning the difference corresponds to about one fourth of the length of the full image. Some further investigation would be required to fully understand this discrepancy between the image coordinates.

Table 4.6: Comparison of the equatorial coordinates.

Name	Unit	Value, simulation	Value, observation	Δ
RA	Degrees	167.203016	167.384988	0.181972
DEC	Degrees	5.301359	6.339560	1.038201

4.2.4 Analysis by `Astrometry.net`

A useful tool to analyse astronomical images of the sky is `Astrometry.net` [93]. The tool allows the user to upload an image, and it then returns astrometric calibration meta-data, as well as which objects that fall inside the FOV. By using `Astrometry.net`, both the real and simulated images from the second simulation have been analysed. The result from the analysis with the found annotated stars, is seen in Figure 4.24.

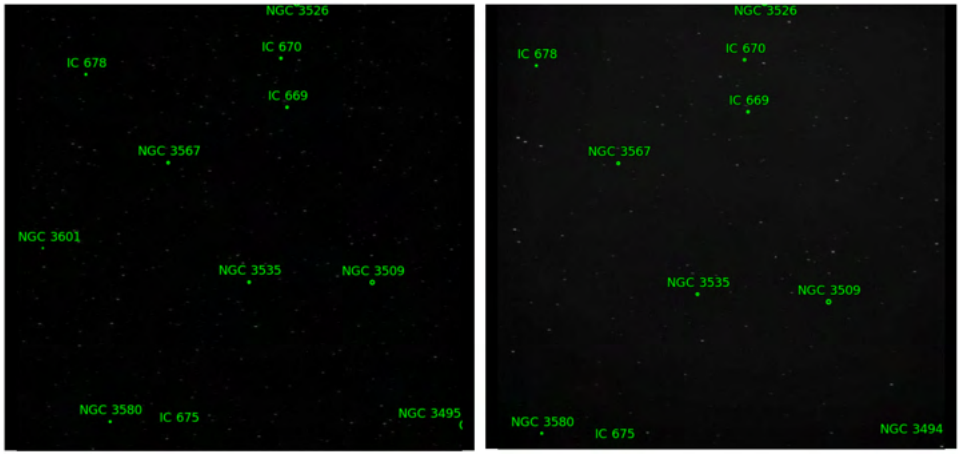


Figure 4.24: Analysis of the images by `Astrometry.net`. The simulated image is located to the left, and the real image is to the right.

As seen, `Astrometry.net` managed to process both the simulated and the real images, and both analyses were successful. This is a strong indicator of that the simulated images can be used for analysis, and possible retrieval of the satellite orbital parameters.

4.3 Python packages dependencies

A pronounced strategy when developing the tool was to use already existing Python packages whenever suitable. The final used packages, which upon the simulation tool depends on, are:

- `Skyfield` - Used for loading and propagating the orbital elements, integrating CelesTrak data, handling positional vectors, loading the ephemeris, loading the Hipparcos catalogue, building the stereographic projection, calculating coordinates, check whether satellite is sunlit, check whether the Sun is up, and applying constellations lines.
- `NumPy` - Used for numerical calculations, building arrays and matrices, conversion between radians and degrees, and several other mathematical operations.
- `Pandas` -Used for creating data frames for the star catalogues, storing satellite coordinates and attributes, handling the large databases, and perform operations on the data frames.
- `Matplotlib` - Used for plotting the figures.
- `AstroPy` - Used for importing constants, handling units, calculating kernels, convolution, and conversion to the FITS file format.

Additionally, both the packages `Poppy` and `Astroquery` have been used in the project. For now, the simulation tool is not dependent on `Poppy`, but code that uses the package has been left in the simulation tool in case the usability of `Poppy` is to be re-examined in the future. The package `Astroquery` has been used to create the star catalogues from GAIA EDR3, but the simulation tool is currently not dependent on the package.

4.4 Evaluation of the fulfilment of the requirements

Section 3.1 presented the requirements which were to be fulfilled by the finished simulation tool. In order to evaluate the simulation tool these will be evaluated in order to see if they have been fulfilled or not.

4.4.1 Requirement group 1: Basic properties

The requirements of the basic properties of the tool were presented in Table 3.1. The simulation tool code has indeed been written in Python, and requirement B-1 is therefore evaluated as fulfilled. Requirement B-2 is more unclear since the visual magnitude of a telescope is not as straightforward to know as anticipated when the requirement was written. Currently, the simulation tool can simulate stars as

faint as magnitude 16. If required, the process of downloading a larger catalogue and have the simulation tool loading it is straightforward. After discussion with the supervisors, it was concluded that the limiting magnitude 17 should be more than enough for the planned location and operations of the telescope. In the reconstruction examples it was also seen that the simulated images displayed more faint tracklets than the real images. This requirement is therefore also evaluated as fulfilled.

Requirement B-3 is also evaluated as fulfilled since the simulation tool can plot both star and satellite tracklets. The simulation tool does take the camera and telescope properties into account, and is therefore also evaluated as fulfilled. Implementing a GUI was a soft requirement, and it was de-prioritised for other tasks. Requirement B-5 is therefore not fulfilled.

4.4.2 Requirement group 2: Input parameters

The requirements regarding the input parameters were presented in Table 3.2. As explained in Section 4.1.1, all these types of inputs have been implemented in the simulation tool. Therefore, all requirements in the group are evaluated as fulfilled.

4.4.3 Requirement group 3: System properties

The requirements that concerned the system properties were presented in Table 3.3. Both camera and telescope parameters are used by the simulation tool, and requirements P-1 and P-2 are therefore both evaluated as fulfilled. Lens deformation has not been investigated, and the soft requirement P-3 is therefore unfulfilled.

4.4.4 Requirement group 4: Disturbances

The requirements related to the different disturbances were presented in Table 3.4. In hindsight, the hard requirement D-1 should have been more specific. However, dark current is a disturbance from thermal noise, and this disturbance has indeed been implemented. This requirement is therefore evaluated as fulfilled. The definition of the atmospheric effects mentioned in requirement D-2 should also have been more specific, but the atmospheric efficiency and effect on the spread of the electrons from seeing have indeed been implemented. This requirement is therefore also evaluated as fulfilled.

The effects from mount disturbances have not been implemented in the tool, and requirement D-3 is therefore not fulfilled. The effects from Moon glow can partially be simulated by the sky background counts parameter, but the Moon often causes a gradient of the sky background which has not been implemented. Requirement D-4 is therefore evaluated as not fulfilled.

The sky background counts parameter is used to simulate the effects of the sky glow, and requirement D-5 is therefore fulfilled. Requirement D-6 concerns dead pixels, which consists of both hot and dark pixels. These have also been

implemented, but the dark current response might need to be adjusted depending on the simulated sensor. This is easily done and implemented in the tool, and requirement D-6 is therefore also evaluated as fulfilled.

4.4.5 Requirement group 5: Satellite observation properties

The last requirement group regarding the satellite observation properties was presented in Table 3.5. This group contains only soft requirements. The modelling of the apparent satellite magnitude includes both the albedo and the area of the satellite, as well as the sun angle. The tool also outputs warnings related to the observation geometry. All requirements in the group are therefore evaluated as fulfilled.

4.4.6 Summary of the requirements evaluation

The fully evaluated requirement matrix can be seen in Table A.3. In summary, all 16 hard requirements have been fulfilled, as well as six soft requirements. Out of the in total 26 requirements, four requirements are evaluated as unfulfilled. All unfulfilled requirements are soft requirements.

Chapter 5

Summary and conclusions

This work aimed at designing and developing a tool to simulate images of satellite passes, as they would look like from a professional telescope. The tool allows the user to choose the observation location, time, telescope, camera, satellite and satellite properties, TLE, disturbances, type of tracking, limiting magnitude, and image format. The code for the tool has been written in Python, and has made use of several packages when suitable.

To evaluate the tool, it has been attempted to replicate two real images of satellite passes. Both of these images were provided by SMARTnet™and obtained by using satellite tracking. The results are that the tool manages well to implement all the basic functions related to time, the orbital parameters, satellite propagation, stars, and telescope and camera properties. More complex functions, such as the seeing and PSF, have also been implemented but with simplifications. The different parameters affects the final resulting image, and SSC should therefore be able to use the tool to examine different observational scenarios in order to plan their upcoming telescope operations.

One of the simulated images has also been analysed by the external software `Astrometry.net`. The software managed to analyse and process the image, as well as extract the names of some stars visible within the FOV. This is a strong indicator that SSC should be able to use the images for training and testing their orbit determination software.

Out of the 26 defined requirements on the tool, all 16 hard requirements were fulfilled. Additionally, six soft requirements were also fulfilled. This concludes that in total 22 out of the 26 requirements were fulfilled.

5.1 Limitations

An important strategic decision to finish the tool within the time limits was to use pre-existing libraries when suitable. This method comes with some limitations, and

as a result the rotation of the FOV is yet to be fully understood. Additionally, the model of the seeing and PSF is simplified. This means that telescope obstruction and diffraction spikes are not included in the images.

The current version of the tool has a highest possible limiting magnitude of 16. Increasing the magnitude has major consequences on the run time for the tool. The run time is further affected by the FOV, exposure time, and chosen Δt.

5.2 Future work

If the simulation tool is to be extended and developed further, these are some areas that can serve as starting points for the future work.

5.2.1 Visibility effects on the satellite tracklet

In the current version of the tool, the effects from eclipses and the satellite position relative to Earth have not been modelled into the appearance of the satellite tracklet. Instead, warnings are outputted in the console window if any of the three visibility demands is unfulfilled during the observation period. Depending on how the tool is to be used, this method might need to be re-evaluated.

5.2.2 Telescope mounting and rotation of FOV

The discrepancy between the rotation of the FOV between the real images and the simulated images requires further investigation. The recommended way is to rewrite the stereographic projection function in order to fully understand it. This would also allow the implementation of different telescope mounts and slewing movement of the telescope.

5.2.3 Improvement of seeing and PSF modelling

The current implementation of the seeing and PSF utilizes simplifications. The modelling of these effects could be improved. There are also remaining uncertainties regarding the sizes of the kernels, and the associated boundary effects. In addition, the effects from diffraction spikes should also be implemented.

These improvements could be initiated by investigating using the Moffat distribution for the seeing kernel. Dr. Flavio Calvio, at Stockholm University, has previously worked with modelling PSFs. He has provided access to his Python code which is found at `https://github.com/calvofl0/psftool`. Due to time constraints it was not possible to implement his code into the simulation tool. This could therefore be a good place to start investigations of an improved PSF.

5.2.4 Filter selection as an user input

Currently, the observed wavelength has been hard coded to 700 nm. Simplifications are also used when modelling the signal, since ideally integration should be performed over the passband spectra. Further investigations are required to understand how to expand the star catalogues to allow filter selection by the user.

5.2.5 Investigate error from overlapping star catalogues

In the current design of the simulation tool it uses two different star catalogues. This means that there is some overlap between the catalogues, which may cause some stars to appear brighter than expected. This overlap should be better understood, and the Hipparcos catalogue could then be modified to decrease the overlap.

5.2.6 Implementation of SNR for advanced imaging analysis

An idea originally intended to implemented regarded the estimation of the SNR of the simulated image. The theory behind this was discussed in Section 2.2.7 and resulted in equation (2.9). The idea was that if the simulation tool was to calculate this parameter, it could be used to learn more about of how much noise could be tolerated to still perform an acceptable observation. However, SNR was not included in the requirements and was therefore de-prioritised to other functionalities of the simulation tool. A longer discussion related to how the images will be used would also be required to understand how to best implement this parameter. Equation (2.9) presented how to calculate the SNR for a single pixel, but it could also be interesting to calculate the SNR for the whole image which would require additional investigation.

5.2.7 Additional and improved functions

Due to time constraints, some functions have been simplified or omitted. Examples of omitted functions are the inclusion of planets and the Moon in the simulated FOV, as well as using a gain parameter when modelling the sensor readout.

Another interesting function to add would be to simulate multiple satellites in the same FOV. This function is not too complicated to implement, but has been de-prioritised in favour of other functionalities. During the discussions with several astronomers, it was also mentioned that it might be desirable to be able to de-focus the telescope to perform the observations of the satellites. This is also a function that could be implemented in the future.

Finally, it needs to be further investigated exactly how the outputs of the tool are to be used in the future. For now, the FITS files are of greatest use to the user, while the PNG files do not display many details. It could therefore be beneficial to scale these images before saving them, but this requires further investigation.

Bibliography

[1] Space debris by the numbers, `www.esa.int/Safety_Security/Space_Debris/Space_debris_by_the_numbers`, 2021, Accessed: 2021-02-12.

[2] T. Schildknecht, *Optical surveys for space debris*, The Astronomy and Astrophysics Review **14**, 41 (2007).

[3] D. J. Kessler and B. G. Cour-Palais, *Collision frequency of artificial satellites: The creation of a debris belt*, Journal of Geophysical Research: Space Physics **83**, 2637 (1978).

[4] J. A. Kennewell and B. N. Vo, *An overview of space situational awareness*, Proceedings of the 16th International Conference on Information Fusion , 1029 (2013).

[5] SSA Programme overview, `www.esa.int/Safety_Security/SSA_Programme_overview`, Accessed: 2021-02-15.

[6] A. Pastor *et al.*, *Correlation Techniques for Catalogue Build-up and Maintenance With Radar and Optical Survey Measurements*, LPI Contributions **2109**, 6098 (2019).

[7] F. Santoni, F. Piergentili and R. Kawashima, *Global University Space Debris Observation Network (GUSDON)*, 8th Space Debris Workshop, 3-5 December 2018, JAXA (2018).

[8] S. H. Hossein *et al.*, *Sapienza Space debris Observatory Network (SSON): A high coverage infrastructure for space debris monitoring*, Journal of Space Safety Engineering **7**, 30 (2020).

[9] J. Cooper, I. Juvan-Beaulieu and A. Wise, *Scientific CMOS Optical Detectors for Orbital Debris Observations*, First International Orbital Debris Conference (2019).

[10] P. Bely, *The design and construction of large optical telescopes* (Springer, 2003).

[11] H. Karttunen *et al.*, *Fundamental astronomy* (Springer, 2017).

[12] Telescope Aperture Explained – Is Bigger Aperture Better?, `www.stargazingpro.com/telescope-aperture-explained/`, Accessed: 2021-02-17.

[13] Newtonian Telescope, `https://en.wikipedia.org/wiki/File:Newtonian_telescope2.svg`, Accessed: 2021-03-22.

[14] Cassegrain Telescope, `https://en.wikipedia.org/wiki/File:Cassegrain_Telescope.svg#metadata`, Accessed: 2021-03-08.

[15] Astrograph Ltd., Corrected Dall Kirkham, `https://astrograph.net/Equipment/Telescopes/Maksutov-Newtonians`, Accessed: 2021-05-22.

[16] I. S. McLean, *Electronic imaging in astronomy: detectors and instrumentation* (Springer Science & Business Media, 2008).

[17] C. R. Kitchin, *Astrophysical Techniques* (CRC Press, 2008).

[18] H. Raab, *Detecting and measuring faint point sources with a CCD*, Proceedings of Meeting on Asteroids and Comets in Europe (MACE) (2002).

[19] F. R. Chromey, *To measure the sky: an introduction to observational astronomy* (Cambridge University Press, 2016).

[20] Wikipedia, Airy Pattern, `https://en.wikipedia.org/wiki/Airy_disk#/media/File:Airy-pattern.svg`, Accessed: 2021-06-03.

[21] Comparison strut diffraction spikes, `https://en.wikipedia.org/wiki/File:Comparison_strut_diffraction_spikes.svg`, Accessed: 2021-03-24.

[22] Understanding Focal Length, `www.nikonusa.com/en/learn-and-explore/a/tips-and-techniques/understanding-focal-length.html`, Accessed: 2021-02-19.

[23] R. D. Coder and M. J. Holzinger, *Multi-Objective Design of Optical Systems for Space Situational Awareness*, Acta Astronautica **128**, 669 (2016).

[24] H. Fiedler *et al.*, *SMARTnet: First Experience of Setting up a Telescope System to Survey the Geostationary Ring*, (2015).

[25] P. Danek, Alt-Azimuth Mount vs Equatorial Mount - What To Choose?, `www.telescopeguides.com/alt-azimuth-mount-vs-equatorial-mount-what-to-choose/`, Accessed: 2021-03-22.

[26] Andor, CCD Blooming and Anti-Blooming, `https://andor.oxinst.com/learning/view/article/ccd-blooming-and-anti-blooming`, Accessed: 2021-03-24.

[27] ProteinSimple, A Guide to CCD Camera Parameters: Applications for Gel and Membrane Imaging, https://offers.the-scientist.com/hubfs/downloads/TS/ProteinSimple/Imaging/Guide_to_CCD_Camera_Parameters.pdf, Accessed: 2021-03-29.

[28] Oxford Instruments, CCD Blemishes Explained?, https://andor.oxinst.com/learning/view/article/ccd-blemishes-and-non-uniformities, Accessed: 2021-03-29.

[29] Oxford Instruments, Understanding Read Noise in sCMOS Cameras, https://andor.oxinst.com/learning/view/article/understanding-read-noise-in-scmos-cameras, Accessed: 2021-03-29.

[30] T. Legault, *Astrophotography* (Rocky Nook, Inc., 2014).

[31] Gaia EDR3 passbands, www.cosmos.esa.int/web/gaia/edr3-passbands, Accessed: 2021-03-23.

[32] Gaia Collaboration, T. Prusti *et al.*, *The Gaia mission*, Astronomy & Astrophysics **595**, A1 (2016), 1609.04153.

[33] Gaia Collaboration, A. G. A. Brown *et al.*, *Gaia Early Data Release 3: Summary of the contents and survey properties*, arXiv e-prints , arXiv:2012.01533 (2020), 2012.01533.

[34] R. L. Cognion, *Large phase angle observations of GEO satellites*, **8739**, 87390K (2013).

[35] R. L. Cognion, *Observations and modeling of GEO satellites at large phase angles*, AMOS Proceedings (2013).

[36] E. F. Knott, J. F. Schaeffer and M. T. Tulley, *Radar cross section* (SciTech Publishing, 2004).

[37] E. M. Vallerie, *Investigation of photometric data received from an artificial earth satellite*, (1963).

[38] D. A. Vallado, *Fundamentals of astrodynamics and applications* (Springer Science & Business Media, 2013).

[39] M. Gallaway, *An introduction to observational astrophysics* (Springer, 2016).

[40] Wikipedia, Azimut altitude, https://commons.wikimedia.org/wiki/File:Azimut_altitude.svg, Accessed: 2021-04-14.

[41] European Space Agency, Gaia Early Data Release 3 (GAIA EDR3), https://www.cosmos.esa.int/web/gaia/earlydr3, Accessed: 2021-04-12.

[42] Université de Strasbourg/CNRS, What is SIMBAD?, http://simbad.u-strasbg.fr/guide/simbad.htx, Accessed: 2021-04-12.

[43] Stellarium, Stellarium Web, https://stellarium-web.org/, Accessed: 2021-05-24.

[44] Cartes du Ciel, SkyChart / Cartes du Ciel - Free software to draw sky charts, https://www.ap-i.net/skychart/en/start, Accessed: 2021-05-24.

[45] Google, Google Sky, https://www.google.com/sky/, Accessed: 2021-05-24.

[46] Chris Peat, Heavens Above, https://www.heavens-above.com/, Accessed: 2021-05-24.

[47] The Astropy Project, The Astropy Project, https://www.astropy.org/, Accessed: 2021-05-24.

[48] Adam Ginsburg, Astroquery, https://astroquery.readthedocs.io/en/latest/, Accessed: 2021-05-24.

[49] Space Telescope Science Institute, Documentation for POPPY, https://poppy-optics.readthedocs.io/en/latest/index.html, Accessed: 2021-06-03.

[50] European Space Agency, Pyxel - A general imaging detector simulation framework in Python for Astronomy and Earth observation, https://esa.gitlab.io/pyxel/, Accessed: 2021-05-24.

[51] Wikipedia, Orbit1, https://en.wikipedia.org/wiki/File:Orbit1.svg, Accessed: 2021-04-16.

[52] Space-Track, Basic Description of the Two Line Element (TLE) Format, https://www.space-track.org/documentation#/tle, Accessed: 2021-04-19.

[53] Skyfield, Skyfield - Earth Satellites, https://rhodesmill.org/skyfield/earth-satellites.html, Accessed: 2021-06-07.

[54] D. Vallado *et al.*, *Revisiting spacetrack report# 3*, AIAA/AAS Astrodynamics Specialist Conference and Exhibit , 6753 (2006).

[55] F. R. Hoots and R. L. Roehrich, *Models for propagation of NORAD element sets*, (1980).

[56] Space-Track, Space-Track.org, https://www.space-track.org/, Accessed: 2021-05-24.

[57] Dr. T.S. Kelso, CelesTrak, http://celestrak.com/NORAD/elements/, Accessed: 2021-05-24.

[58] Skyfield, Skyfield - Elegant astronomy for Python, https://rhodesmill.org/skyfield/, Accessed: 2021-06-03.

[59] B. Rhodes, Skyfield: High precision research-grade positions for planets and Earth satellites generator, 2019, 1907.024.

[60] CS GROUP, Orekit -An accurate and efficient core layer for space flight dynamics applications, https://www.orekit.org/, Accessed: 2021-06-03.

[61] AGI, Systems Tool Kit (STK), https://www.agi.com/products/stk, Accessed: 2021-06-03.

[62] Skyfield, Skyfield - Example Plots, https://rhodesmill.org/skyfield/example-plots.html, Accessed: 2021-06-06.

[63] Skyfield, Skyfield - Planets and their moons: JPL ephemeris files, https://rhodesmill.org/skyfield/planets.html, Accessed: 2021-06-06.

[64] N. H. Lee, *Geometry: from isometries to special relativity* (Springer, 2020).

[65] Gaia Collaboration, Gaia, GEDR3, Catalogue sizes, http://cdn.gea.esac.esa.int/Gaia/gedr3/_catalogue_sizes.txt, Accessed: 2021-06-07.

[66] Python Software Foundation, Pickle — Python object serialization, https://docs.python.org/3/library/pickle.html, Accessed: 2021-06-07.

[67] Skyfield, Skyfield API Reference — Astronomical Positions, https://rhodesmill.org/skyfield/api-position.html#skyfield.positionlib.ICRF.separation_from, Accessed: 2021-06-08.

[68] Skyfield, Skyfield - Find when a satellite is in sunlight, https://rhodesmill.org/skyfield/earth-satellites.html#satellite-is-sunlit, Accessed: 2021-06-09.

[69] Skyfield, Skyfield - Sunrise and Sunset, https://rhodesmill.org/skyfield/almanac.html#sunrise-and-sunset, Accessed: 2021-06-09.

[70] Photutils Developer, Photutils - Building an effective Point Spread Function (ePSF), https://photutils.readthedocs.io/en/stable/epsf.html#build-epsf, Accessed: 2021-06-13.

[71] Space Telescope Science Institute, POPPY - Examples, https://poppy-optics.readthedocs.io/en/stable/examples.html#a-simple-circular-pupil, Accessed: 2021-06-13.

[72] The Astropy Developers, Astropy - convolve, https://docs.astropy.org/en/stable/api/astropy.convolution.convolve.html#astropy.convolution.convolve, Accessed: 2021-06-13.

[73] The Astropy Developers, Astropy - convolve_fft, https://docs.astropy.org/en/stable/api/astropy.convolution.convolve_fft.html, Accessed: 2021-06-13.

[74] The Astropy Developers, Convolution and filtering, https://het.as.utexas.edu/HET/Software/Astropy-0.4.2/convolution/index.html, Accessed: 2021-06-13.

[75] The Astropy Developers, AiryDisk2DKernel, https://docs.astropy.org/en/stable/api/astropy.convolution.AiryDisk2DKernel.html#airydisk2dkernel, Accessed: 2021-06-13.

[76] The Astropy Developers, Gaussian2DKernel, https://docs.astropy.org/en/stable/api/astropy.convolution.Gaussian2DKernel.html#astropy.convolution.Gaussian2DKernel, Accessed: 2021-06-13.

[77] The Astropy Developers, Moffat2DKernel, https://docs.astropy.org/en/stable/api/astropy.convolution.Moffat2DKernel.html#astropy.convolution.Moffat2DKernel, Accessed: 2021-06-13.

[78] M. Craig and L. Chambers, CCD Reduction Guide - Construction of an artificial (but realistic) image, https://mwcraig.github.io/ccd-as-book/01-03-Construction-of-an-artificial-but-realistic-image.html, Accessed: 2021-06-11.

[79] The SciPy community, NumPy - numpy.random.Generator.poisson, https://numpy.org/doc/stable/reference/random/generated/numpy.random.Generator.poisson.html#numpy.random.Generator.poisson, Accessed: 2021-06-10.

[80] The SciPy community, NumPy - numpy.random.Generator.normal, https://numpy.org/doc/stable/reference/random/generated/numpy.random.Generator.normal.html#numpy.random.Generator.normal, Accessed: 2021-06-11.

[81] The SciPy community, NumPy - numpy.random.Generator.integers, https://numpy.org/doc/stable/reference/random/generated/numpy.random.Generator.integers.html#numpy.random.Generator.integers, Accessed: 2021-06-11.

[82] The Astropy Developers, FITS File Handling, https://docs.astropy.org/en/stable/io/fits/index.html, Accessed: 2021-06-12.

[83] ESA/ESO/NASA, FITS Liberator, https://noirlab.edu/public/products/applications/app002/, Accessed: 2021-06-12.

[84] H. Fiedler *et al.*, *SMARTnet™–First Results of the Telescope Network*, development **1**, 2 (2017).

[85] Time and Date, Current Local Time in Cape Town, South Africa, https://www.timeanddate.com/worldclock/south-africa/cape-town, Accessed: 2021-06-14.

[86] Baader Planetarium, FLI ProLine CCD camera KAF-16803, https://www.baader-planetarium.com/en/fli-proline-kaf-16803-ccd-camera.html, Accessed: 2021-06-14.

[87] PlaneWave Instruments, CDK20 OPTICAL TUBE ASSEMBLY (F/6.8), https://planewave.com/product/cdk20-ota/, Accessed: 2021-06-14.

[88] N2YO.com, NAVSTAR 65 (USA 213), https://www.n2yo.com/satellite/?s=36585, Accessed: 2021-06-14.

[89] Humboldt State University, Atmospheric Absorption & Transmission, http://gsp.humboldt.edu/OLM/Courses/GSP_216_Online/lesson2-1/atmosphere.html, Accessed: 2021-06-15.

[90] N2YO.com, INTELSAT 901, https://www.n2yo.com/satellite/?s=26824, Accessed: 2021-06-17.

[91] ASA Astrosysteme GmbH, ASA PRODUCT FOLDER, https://www.skypoint.it/img/cms/Cataloghi/ASA_catalogue.pdf, Accessed: 2021-06-17.

[92] Time and Date, Moonrise, Moonset, and Phase Calendar for Cape Town, March 2020, https://www.timeanddate.com/moon/south-africa/cape-town?month=3&year=2020, Accessed: 2021-06-17.

[93] Astrometry.net, Astrometry.net, http://astrometry.net/, Accessed: 2021-06-18.

Appendix A

Complete system requirement matrix

The requirement matrix was reviewed and discussed in Chapter 3, and each group of requirements were presented together with their rationales. The legend of the requirement matrix is presented in Table A.1, and the full concatenated requirement matrix is presented in Table A.2. Later, in Section 4.4, the fulfilment of the requirements was discussed. The evaluated requirement matrix is seen in Table A.3.

Table A.1: Legend of the requirement matrix.

Legend	
B	Basic properties
I	Input parameters
P	System properties
D	Disturbances
S	Satellite observation properties
"Shall" is used to indicate a hard requirement, meaning it must be implemented. "Should" is used to indicate a soft goal which is desirable, but not formally required.	

Table A.2: The full concatenated requirement matrix.

ID	Requirement	Verification Method	Rationale
B-1	The code shall be written in Python.	Inspection	
B-2	The tool shall be able to simulate stars down to the visual magnitude limit of the telescope parameters.	Analysis	
B-3	The code shall be able to plot a satellite tracklet.	Inspection	Objective
B-4	The system parameters shall include camera and telescope properties.	Test	Objective
B-5	The tool should have a graphical user interface.	Inspection	Facilitate user interaction
I-1	The tool shall be able to recieve geographic coordinates as input.	Test	In order to test different observation sites
I-2	The tool shall be able to recieve observation altitude as input.	Test	
I-3	The tool shall be able to recieve time as input.	Test	
I-4	The user shall be able to choose between sidereal or satellite tracking.	Test	
I-5	The tool shall be able to recieve exposure time as input.	Test	Will affect length of tracklet.
I-6	The tool shall be able to import satellite orbital information.	Test	
I-7	The tool shall be able to recieve system parameters.	Test	Sensor size, pixel size, noise figure.
I-8	The tool shall be able to recieve binning as an input.	Test	In order to test different resolutions.
I-9	The tool shall be able to recieve image format as input.	Test	
P-1	The tool shall model the image based on camera properties.	Test	Sensor size, pixel size, noise figure.
P-2	The tool shall model the image based on telescope properties.	Test	Aperture, focal length, obstruction, mirror/lense type and properties.
P-3	The tool should model lens deformation of the telescope.	Analysis	
D-1	The tool shall model thermal noise.	Analysis	
D-2	The tool should model atmospheric effects.	Analysis	Winds, jet streams, upmoving air, transparency.
D-3	The tool should model mount disturbances.	Analysis	
D-4	The tool should model the effect of moon glow.	Analysis	
D-5	The tool should model the effect of sky glow.	Analysis	
D-6	The tool should model dead pixels.	Test	
S-1	The tool should take the albedo of the satellite into account.	Test	
S-2	The tool should take the area of the satellite into account.	Test	
S-3	The tool should take the observation geometry into account, i.e. sun angles.	Analysis	Will affect shadowing and satellite visibility

Table A.3: The evaluated requirement matrix.

ID	Requirement	Fulfilled	Unfulfilled
B-1	The code shall be written in Python.	X	
B-2	The tool shall be able to simulate stars down to the visual magnitude limit of the telescope parameters.	X	
B-3	The code shall be able to plot a satellite tracklet.	X	
B-4	The system parameters shall include camera and telescope properties.	X	
B-5	The tool should have a graphical user interface.		X
I-1	The tool shall be able to recieve geographic coordinates as input.	X	
I-2	The tool shall be able to recieve observation altitude as input.	X	
I-3	The tool shall be able to recieve time as input.	X	
I-4	The user shall be able to choose between sidereal or satellite tracking.	X	
I-5	The tool shall be able to recieve exposure time as input.	X	
I-6	The tool shall be able to import satellite orbital information.	X	
I-7	The tool shall be able to recieve system parameters.	X	
I-8	The tool shall be able to recieve binning as an input.	X	
I-9	The tool shall be able to recieve image format as input.	X	
P-1	The tool shall model the image based on camera properties.	X	
P-2	The tool shall model the image based on telescope properties.	X	
P-3	The tool should model lens deformation of the telescope.		X
D-1	The tool shall model thermal noise.	X	
D-2	The tool should model atmospheric effects.	X	
D-3	The tool should model mount disturbances.		X
D-4	The tool should model the effect of moon glow.		X
D-5	The tool should model the effect of sky glow.	X	
D-6	The tool should model dead pixels.	X	
S-1	The tool should take the albedo of the satellite into account.	X	
S-2	The tool should take the area of the satellite into account.	X	
S-3	The tool should take the observation geometry into account, i.e. sun angles.	X	

Appendix B

Complementary results from the first reconstruction

B.1 Resulting sensor without applied spread

In Section 4.1.2, the outputted image of the reconstruction simulation was showed in Figure 4.7. The corresponding image from a simulation run without applying any type of spread, is seen in Figure B.1. This time the colour bar maximum value is 100,000, which equals the well depth of the sensor. This indicates that the image contains saturated pixels. The star tracklet which was seen in Figure 4.7 can hardly be discerned since the electrons are now contained to much fewer pixels in total.

B.2 Resulting PNG file

The saved PNG file of the simulated sensor is shown in Figure B.2. This is a stripped version of the simulated sensor figure, which is directly outputted from the simulation tool. When saving the sensor as a PNG file, an associated text log file is created and saved too. This file is seen in Figure B.3.

B.3 Complementary FITS results

The reconstructed sensor simulation file is saved as a FITS file, where the header contains most of the user input. A snippet of the FITS header, as presented when viewed in `ESA/ESO/NASA FITS Liberator`, is seen in Figure B.4. An image zoomed in on some of the star tracklets of the simulated sensor was presented in Figure 3.6. The exact location of this zoomed in version, compared to the complete image, is seen in Figure B.5.

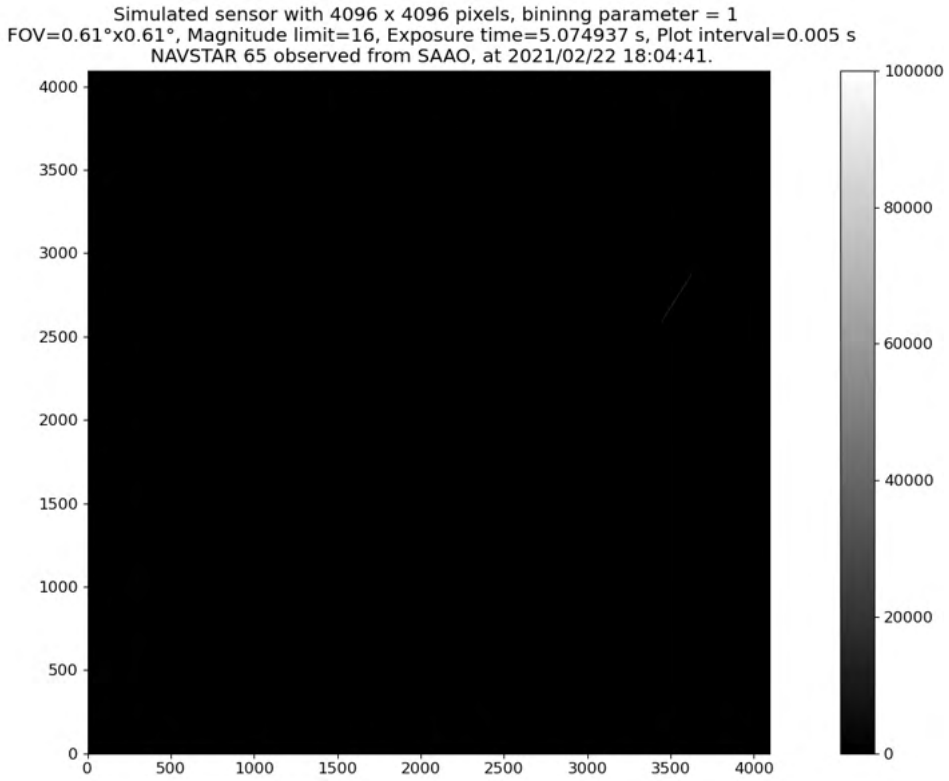


Figure B.1: The corresponding outputted sensor image from the reconstruction simulation, without any spread of electrons applied. This sensor contains saturated pixels.

Figure B.2: The resulting PNG image from the reconstruction simulation. This image is saved directly to the computer, together with an associated text log file containing important user inputs.

```
DATE = 2021/02/22
TIME = 18:04:41 [UTC]
EXPTIME = 5.074937 [seconds]
DELTA T= 0.005 [seconds]
OBJECT = NAVSTAR 65 (USA 213 - GPS 25- NORAD 36585 - 10022A)
ALBEDO = 0.2
RCS = 6.4686 [meter squared]
RA = 8.03 hours
DEC = -3.00 degrees
DISTANCE = 21831.60 km
LOCATION = South African Astronomical Observatory
LOCLAT = -32.38072
LOCLONG = 20.81078
FOV = 0.61x0.61 degrees
DIAMETER = 0.508 [m]
AREAEFF = 0.17 [m^2]
FOCALLEN = 3.454 [m]
FOCALNUM = 6.80
MAXDEPTH = 100000.0 [electrons]
PIXEL_W = 9e-06 [m]
PIXEL_H = 9e-06 [m]
SENSOR_W = 4096 [number of pixels]
SENSOR_H = 4096 [number of pixels]
QUANTEFF = 0.85
DARKCURR = 2.98e-10 [electrons/pixel/second]
WAVELEN = 7e-07 [m]
PASSBAND = 2e-07 [m]
READNOIS = 10.0 [electrons]
DIFFLIM = 1.6811023622047242e-06 [radians]
BINPARAM = 1 [number of pixels per side in bin]
SIGMA = 0.5
```

Figure B.3: The resulting text log file, which is created together with the PNG file. It contains most of the user inputs.

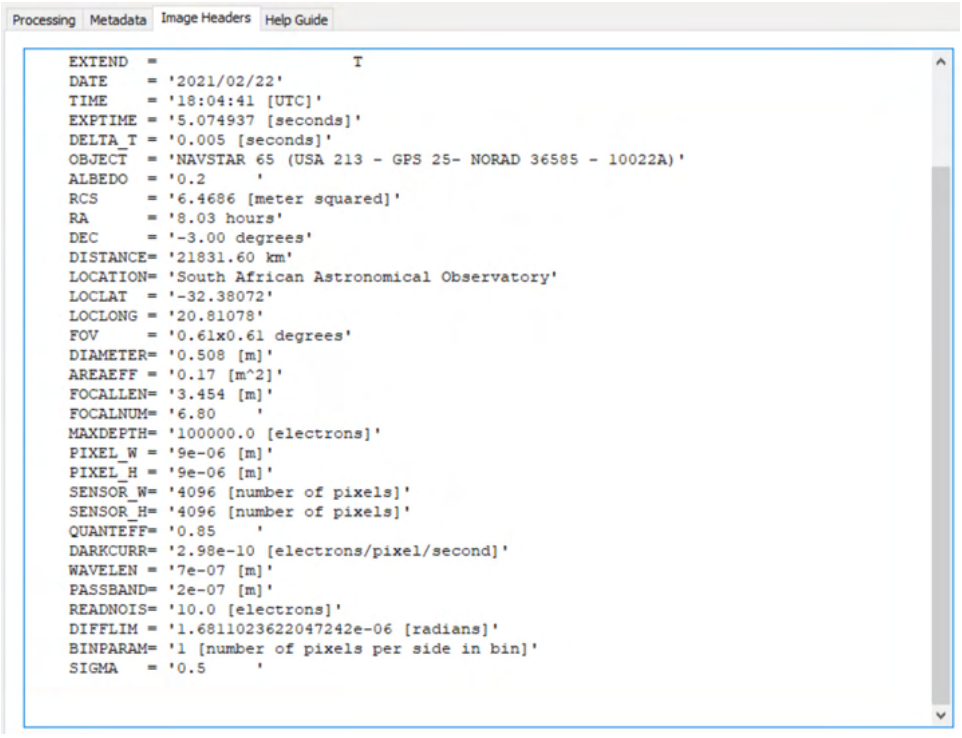


Figure B.4: A snippet of the FITS header, as presented when viewed in `ESA/ESO/NASA FITS Liberator`.

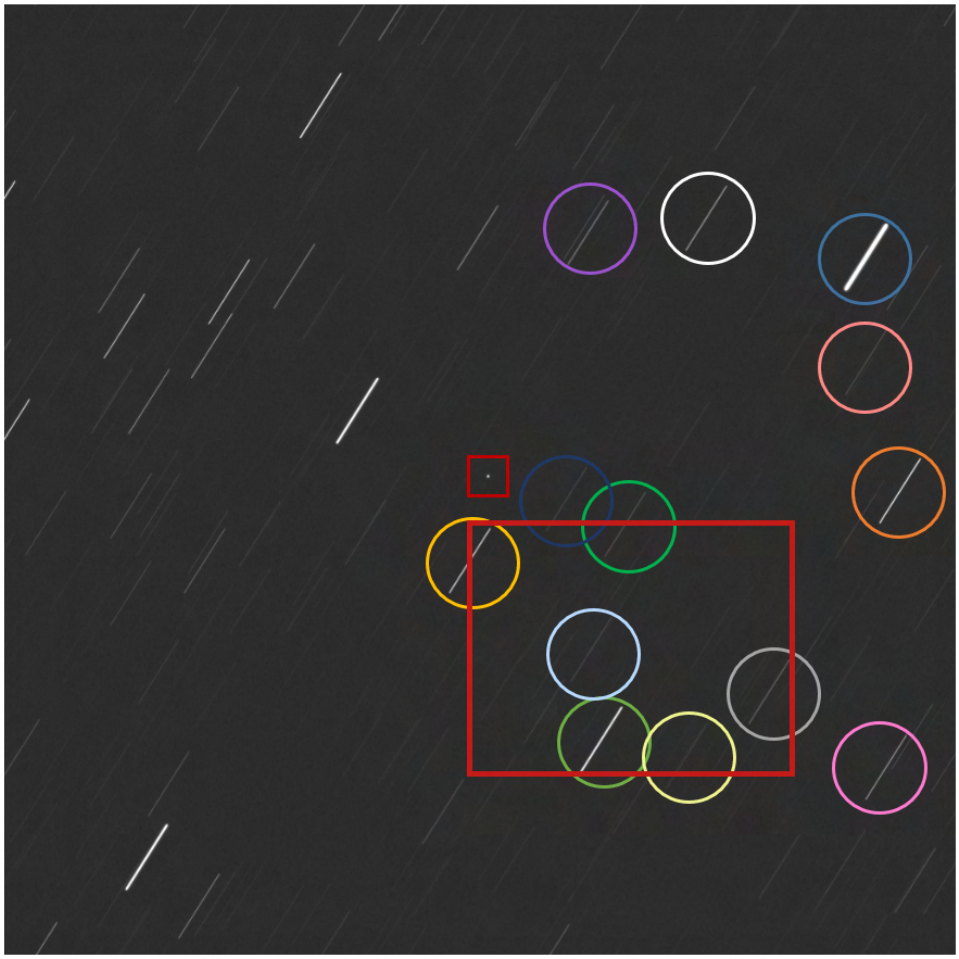


Figure B.5: Image showing the location of the zoomed in image in Figure 3.6. The location is marked by the big red square.

Appendix C

Complementary results from the second reconstruction

C.1 Complementary outputs from the second reconstruction simulation

Section 4.2 discusses the reconstruction of an image acquired from observing the geostationary satellite INTELSAT 901. The original SMARTnet™image was seen in Figure 4.16. The corresponding simulated sensor, which is directly outputted from the simulation tool, is seen in Figure C.1. The corresponding help image is seen in Figure C.2, and the PNG image which is saved directly to the computer is seen in Figure C.3.

C.2 Tracklets comparison cutout

In Section 4.2.3, a zoomed in image of some star tracklets were shown in Figure 4.23. The location of the zoomed in images from the full scale image is seen in Figure C.4, where it has been marked by a pink square.

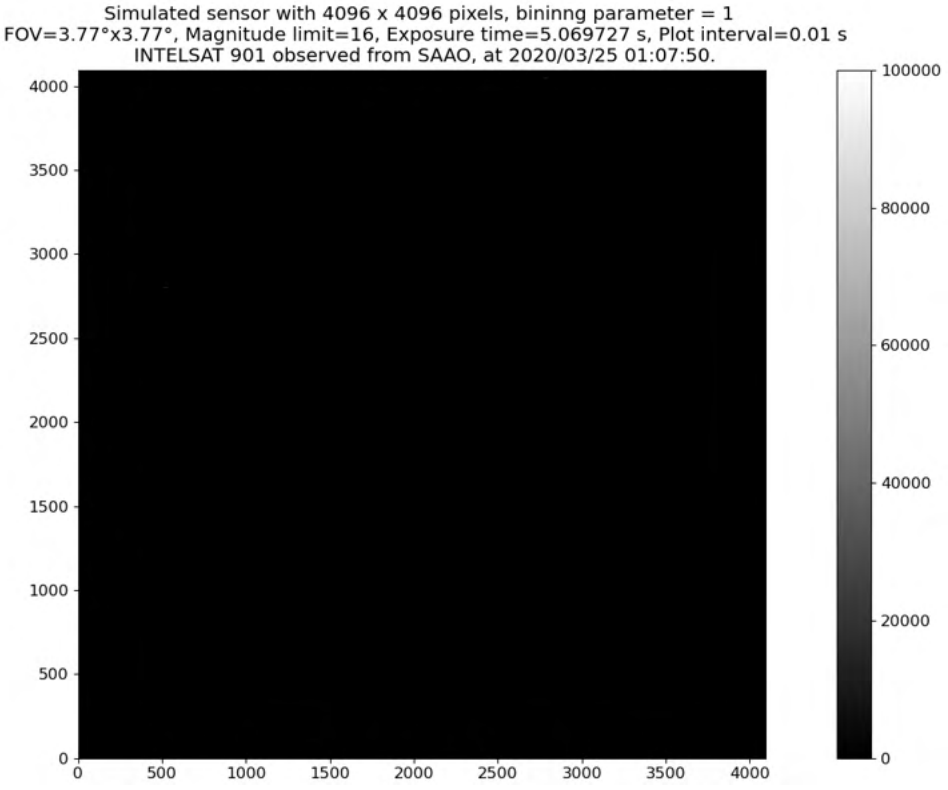


Figure C.1: The directly outputted sensor image from the second reconstruction simulation.

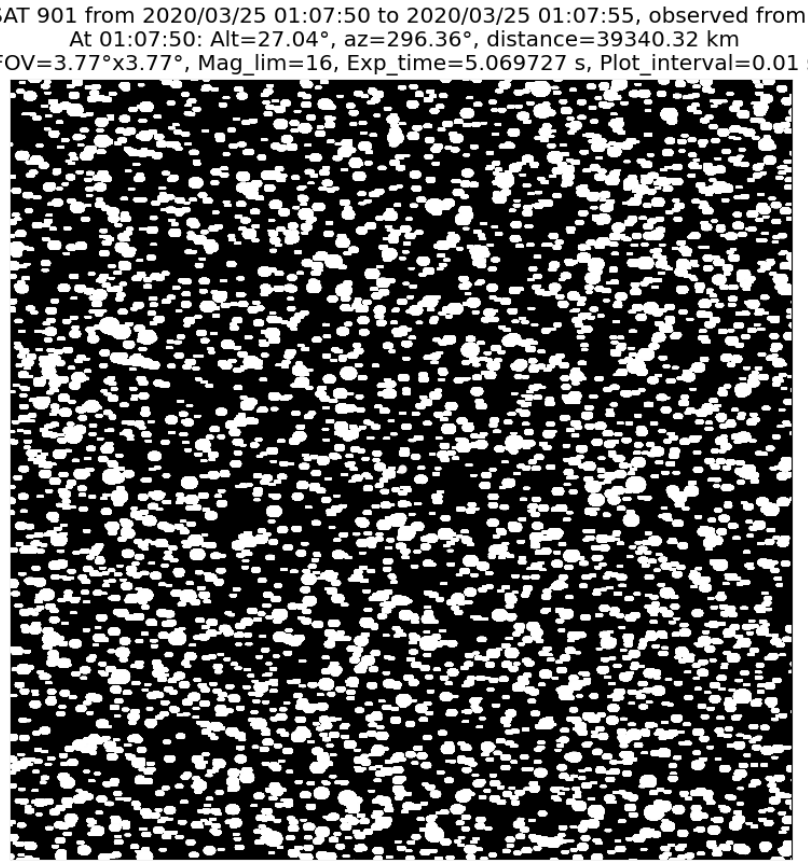


Figure C.2: The outputted help image from the second reconstruction simulation.

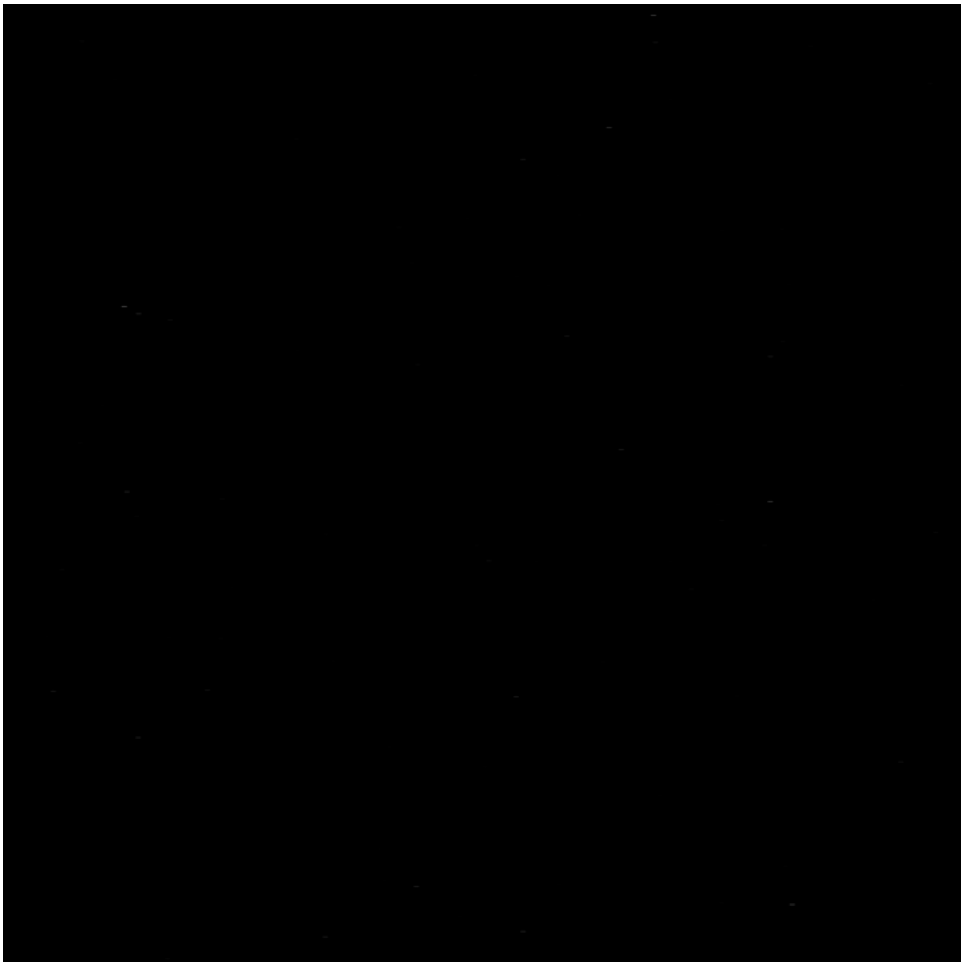


Figure C.3: The resulting PNG file from the second reconstruction simulation.

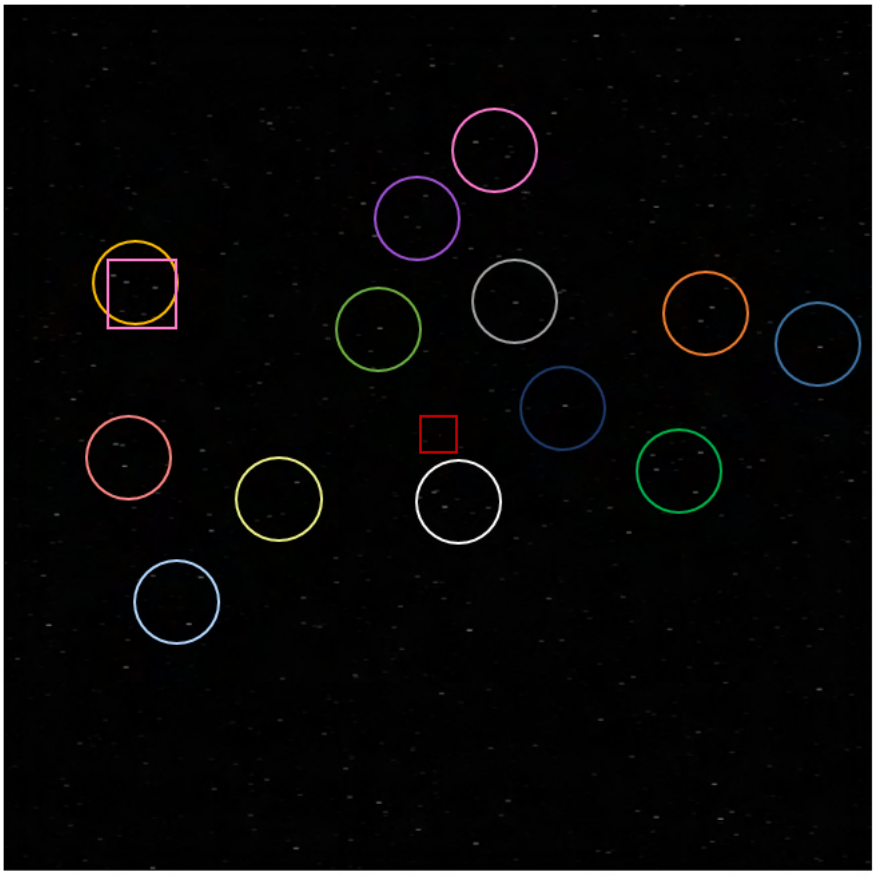


Figure C.4: Image showing the location of the zoomed in image in Figure 4.23. The location is marked by a pink square.

TRITA-EECS-EX-2021:518